AD-A123 772 PROCEEDINGS OF THE 1981 KL-ONE WORKSHOP(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA J G SCHMOLZE ET AL. JUN 82 BBN-4842 N00014-77-C-0378 1/3

UNCLASSIFIED F/G 5/6 NL

MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

Bolt Beranek and Newman Inc.

AD A 123772

Report No. 4842

Proceedings of the 1981 KL-One Workshop

J.G. Schmolze and R.J. Brachman

June 1982

Prepared for:
Advanced Research Projects Agency

DTIC
ELECTE
JAN 26 1983
S D
D

DTIC FILE COPY

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

83 01 26 021

Accession For
NTIS GRA&I ☒
DTIC TAB ☐
Unannounced ☐
Justification

By
Distribution/
Availability Codes

Dist	Avail and/or Special
A	

DTIC COPY INSPECTED 2

Report No. 4842

PROCEEDINGS OF THE 1981 KL-ONE WORKSHOP

James G. Schmolze

Ronald J. Brachman

June 1982

Prepared by:

Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, Massachusetts 02238

Prepared for:

Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

DISTRIBUTION STATEMENT A
Approved for public release;
Distribution Unlimited

Unclassified

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER BBN Report No. 4842	2. GOVT ACCESSION NO. AD-A123 772	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PROCEEDINGS OF THE 1981 KL-ONE WORKSHOP		5. TYPE OF REPORT & PERIOD COVERED Workshop Proceedings 1981
		6. PERFORMING ORG. REPORT NUMBER Report No. 4842
7. AUTHOR(s) James G. Schmolze and Ronald J. Brachman		8. CONTRACT OR GRANT NUMBER(s) N00014-77-C-0378
9. PERFORMING ORGANIZATION NAME AND ADDRESS Bolt Beranek and Newman Inc. 10 Moulton Street Cambridge, MA 02238		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS 7D30
11. CONTROLLING OFFICE NAME AND ADDRESS Office of Naval Research Department of the Navy Arlington, VA 22217		12. REPORT DATE June 1982
		13. NUMBER OF PAGES 269
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) Unclassified
		15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

Distribution of this document is unlimited. It may be released to the Clearinghouse, Department of Commerce, for sale to the general public.

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

artificial intelligence, knowledge representation, semantic networks, structured inheritance networks, KL-ONE.

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The Second KL-ONE Workshop gathered researchers from twenty-one universities and research institutions for a series of discussions and presentations about the KL-ONE knowledge representation language. These proceedings summarize the discussions and presentations, provide position papers from the participants, list the agendas of the Workshop along with the names and addresses of the participants, and include a description of the KL-ONE language plus an index of some KL-ONE technical terms.

DD FORM 1 JAN 73 1473 EDITION OF 1 NOV 65 IS OBSOLETE

Unclassified.

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

Report No. 4842

PROCEEDINGS OF THE 1981 KL-ONE WORKSHOP

James G. Schmolze

Ronald J. Brachman

June 1982

Prepared by:

Bolt Beranek and Newman Inc.
10 Moulton Street
Cambridge, Massachusetts 02238

Prepared for:

Advanced Research Projects Agency
1400 Wilson Boulevard
Arlington, Virginia 22209

(These proceedings are also available from the
Fairchild Laboratory for Artificial Intelligence Research
as FLAIR Technical Report No. 4)

TABLE OF CONTENTS

LIST OF FIGURES

INTRODUCTION

The Second KL-ONE Workshop gathered researchers from twenty one universities and research institutions to the White Mountains for a series of discussions and presentations about the KL-ONE knowledge representation language. The Workshop - held October 16-20, 1981 - was the second to be held at the Christmas Farm Inn in Jackson, New Hampshire. It was an excellent setting for some lively discussions about knowledge representation in general, and our particular experiences with KL-ONE.

While the first Workshop was successful in a modest way, we learned from it that having a large group of people in attendance, especially with disparate goals and backgrounds, was not a way to make technological progress (even the modest amount we had hoped for). On the other hand, simply restricting participation to a very small set of seriously committed researchers seemed equally unsatisfactory, since it did not allow us to report on recent developments to the community at large, nor did it allow that community to get together to share ideas, experiences, and problems. So, this year we opted for a two-part Workshop, the first comprising three days of intensive technical discussions by a small group (14 participants) intimately involved with KL-ONE development, the second comprising two days of presentations, small group discussions, and plenty of free time to knock heads over the issues of the day.

The technical discussions that preceded the main conference covered areas of current central concern to KL-ONE and knowledge representation in general, including "realization" (attributing new descriptions to individuals as they are learned about) and "classification" (putting KL-ONE descriptions into a taxonomy according to their internal structure); Individual Concepts (the way to represent definite descriptions in KL-ONE); "Role Set Relations" (the way to represent constraints in concept definitions in KL-ONE) and "Qua-Concepts" (concepts defined as functions of other concepts); and some system maintenance and utility issues (KL-ONE is implemented in INTERLISP at BBN and Smalltalk at Xerox PARC). To allow us to get right to work, the chairman of each session circulated a position paper to the group in advance, raising the questions he wanted to see addressed at the Workshop.

For the general conference session (attended by 46 people), we invited groups from various sites to report on interesting applications of KL-ONE, problems with it, interesting technical questions, etc. Topics of the talks included "KloneTalk" (the version of KL-ONE implemented in Smalltalk - this included a videotaped demonstration of the system's interface), prototypes in knowledge representation, translation of INTERLISP KL-ONE to FranzLisp, a calculus of Structural Descriptions, and the KL-ONE Classifier, not to mention several others. We also had the larger group break up into smaller working groups to consider inference in KL-ONE, representing beliefs, some KL-ONE practice examples, and transporting KL-ONE to other machines.

All of these topics are covered in these Proceedings.

- o Chapter 1 contains summaries of all of the Technical Discussions that occurred prior to the Main Conference. The authors/editors of these reports have tried to present the problems that were covered, in plain terms, before proceeding into detailed analyses of the discussions.

- o Chapter 2 contains reports on the small group discussions that were held during the Main Conference. Each report contains a topic description and discussion summary.

- o Chapter 3 contains papers for prepared presentations that were made at the Main Conference.

- o Chapter 4 contains positions papers that were submitted by Workshop participants. In some cases, these papers represent the position of an entire research group, while others represent individual opinions.

- o The Appendices contain the agendas and lists of participants for the two Workshop sessions, an address list of members of the KL-ONE community, a summary of KL-ONE for the uninitiated, an index of KL-ONE technical terms, and an index of authors and co-authors. The index of technical terms is extensive, but incomplete - we hope that it is useful to newcomers to KL-ONE.

Please note that there is no list of references for the entire proceedings. Instead, all cited references are listed after each paper and discussion summary.

The reader of these proceedings should note the assumptions made by the authors about the character of the reading audience. In our set of position papers, the assumption of each author may not be explicitly set out, and these vary from paper to paper. However, it is safe to assume that a reader who is somewhat familiar with KL-ONE will find them readily comprehensible. In our summaries of the Technical Discussions, however, we distinctly assumed that our readers would be familiar with the full range of the KL-ONE language (and a fair amount of the jargon used by the community). While this does not suit everyone's needs, the task of transforming all references to KL-ONE objects and ideas into generally accessible language would have delayed these proceedings excessively. We have tried in various ways to make up for the technical nature of these reports: each author was encouraged to open his report with a plain language description of the problem being addressed at his session; we have included a summary of many of the ideas in KL-ONE as an appendix to this document; an index of technical terms is also included. We hope that a significant portion of the material discussed at the Workshop and reported here will be accessible to anyone with a sincere interest in knowledge representation. We encourage your comments and interest.

It should be noted that much of the excitement of the Workshop cannot be captured here - many of the liveliest and most interesting discussions took place spontaneously, outside the range of our microphones. If you are interested in the results of conversations like those, please come and join us at the next Workshop.

Jim Schmolze

Ron Brachman

ACKNOWLEDGEMENTS

Many thanks are given to the participants of this Workshop for their contributions both during the Workshop and to this proceedings. Special thanks are extended to the participants in the technical discussions, as they spent many, many hours preparing for the Workshop, and even more hours preparing the summaries contained in this document (the effort involved is briefly described in the introduction to Chapter 1). Finally, our most sincere thanks goes to Patricia Brinkman for the energy she devoted to the organization of the Workshop and the preparation of this Proceedings.

Although there were no funds provided specifically for the Workshop itself, participation in the Workshop by employees of Bolt Beranek and Newman, plus much of the Workshop's organization, was supported by the Advanced Research Projects Agency of the Department of Defense and was monitored by ONR under the project titled "Research in Knowledge Representation for Natural Language Understanding", contract number N00014-77-C-0378. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of the Defense Advanced Research Projects Agency or the U.S. Government.

1. TECHNICAL DISCUSSIONS

The first three days of the Workshop comprised intensive technical discussions concerning current views and future directions of KL-ONE. These discussions are summarized in this chapter. The reader should note, however, that the positions presented in these summaries will not necessarily be reflected in future research concerning KL-ONE.

As part of the documentation of these discussions, we feel obliged to describe the organization of these sessions, as well as the manner by which we attempted to capture their content. Attendance at this portion of the Workshop was limited to those researchers who were either involved in the development of KL-ONE, or were regular users of KL-ONE. From amongst the possible candidates, fourteen researchers from five institutions attended (they are listed in Appendix B). The choice of topics was made well in advance of the Workshop, as was the assignment of a chairperson for each topic. We also decided to have one session for each topic, which all participants would attend, with strict time limits for each session.

Each chairperson distributed a written description of the topic to be addressed at his session before the Workshop began so that all attendees could prepare for the session, and each chairperson led his own session with the authority to recognize speakers and to direct and/or terminate discussions. In order to help capture the discussion's content, two secretaries per session were assigned (from among the fourteen participants) to take detailed notes, and a tape recording was made of each discussion.

Following the Workshop, each pair of secretaries prepared a summary from their notes and tape recording. The chairpersons then prepared descriptions from these summaries, the recordings, and their own notes. Each participant of the technical discussions was then given the opportunity to suggest edits to these descriptions. Finally, the chairpersons took these suggestions and prepared the final form of the descriptions that are presented in this chapter.

1.1 Assertions in KL-ONE

(including a conceptual reconstruction of the overall structure of the language)

Chaired by Ron Brachman and Hector Levesque (Fairchild)

This particular technical discussion was originally intended to begin paying off the long overdue debt on assertions in KL-ONE. All along, the claim had been made that KL-ONE was somehow different from other representation languages in that it stood firm on the difference between "definitional" and "assertional" information, and had a place and a methodology for dealing with each. Yet most of the work to date had concentrated solely on definitional issues. We intended, then, to use this session as a forum to determine how to "do assertions" in KL-ONE.

However, in a recent attempt to be clear about this distinction and to better understand the overall functionality of KL-ONE, a group of us (Ron Brachman, Richard Fikes, Austin Henderson, Hector Levesque, and, more recently, Dan Bobrow and Mark Stefik) had been re-examining a number of the fundamental characteristics of the language. By the time we arrived at the Workshop, we felt compelled to start our discussion with a conceptual re-construction of KL-ONE, since our position on assertions had become intimately tied to our overall perception of the language. This report begins with an explanation of that reconstruction (just as the Workshop session did), and then addresses some of the issues discussed at the Workshop.

Cleaning up KL-ONE

KL-ONE, as we see it, can be divided into two major components, which we will call the Terminological Component (or Tbox) and the Assertional Component (or Abox). Under the current regime, the Tbox would contain entities like Concepts, Roles, and Structural Descriptions while the Abox would contain Nexuses, Contexts, and Description Wires. Roughly speaking, the Tbox is intended to maintain an evolving language and understand the relationships between its components as purely linguistic expressions. For example, it would be part of the competence of the Tbox to be able to determine when one Concept subsumed another solely on the basis of the Concepts' internal structure. The Abox, on the other hand, is intended to maintain an evolving picture of a world and understand the relationships between its components as assertions about that world. So it might be part of the competence of the Abox to realize when the existence of a Nexus in a Context implied the existence of other Nexuses in that Context.

To a certain extent, all of this is already part of the lore of the KL-ONE research community. It is our contention, however, that the components of the Tbox have on more than one occasion been (mis)used as if they belonged to the Abox, thereby blurring the distinction we speak so strongly about; further, the treatment of the Abox in KL-ONE has been superficial and haphazard. Finally, what KL-ONE as an abstract framework was supposed to do has been confused with issues involved in implementations.

Consider, for example, a "traditional" KL-ONE network with a Concept labelled ELEPHANT having two subConcepts called INDIAN-ELEPHANT and AFRICAN-ELEPHANT. It is very tempting in this situation to answer a question like "How many kinds of elephants are there?" by counting the number of subConcepts below ELEPHANT and, therefore, replying, "Two". The trouble here is that the KL-ONE taxonomy is also thought of as a virtual lattice where there are an infinite number of subConcepts below any given one. For example, in this sense, there are an infinite number of kinds of elephants, namely, CIRCUS-ELEPHANTS, SMALL-ELEPHANTS, MALE-ELEPHANTS, SMALL-MALE-ELEPHANTS and so on (i.e., terms for all of the imaginable kinds of elephants, regardless of whether or not any actually exist). Our feeling is that the attempt to say that there are only two biological kinds of elephants by having only two subConcepts of ELEPHANT is a confusion between uses of the Tbox and Abox. Saying anything at all about the world (as in how many kinds of elephants there are) is the province of the Abox; the Tbox itself cannot be used for such assertions - except to provide the relevant linguistic terms.

So, then, how are we to interpret a "traditional" KL-ONE network in which a Concept like "elephant" has exactly two (not three, not an infinite number of) subConcepts? Our feeling is that the way to understand this is precisely the same as the way to understand the functionality of atoms in LISP. LISP gives you the ability to act as if every atom existed, by creating a new data structure for one on first mention. Given this infinitely generative functionality, how is it that the function *MAPATOMS* can ever halt? We realize that mentioning an atom and the function MAPATOMS are really at two different "levels". MAPATOMS is in fact a function over the data structures that are used to implement the notion of atoms. In very much the same way, a function that gets hold of the subConcepts of a Concept is a *function over* the data structures used to implement KL-ONE, and not one that works on the Tbox or Abox. If we call the network language that we have used to implement the functionality of KL-ONE "SI-Nets" (for "Structured Inheritance Nets"), then fetching the "subConcepts" of a *Concept* is an SI-Net notion that involves fetching data structures connected to other data structures in certain ways. That the data structures have names like "Concept" is of no consequence.

The SI-Net level is the level at which the data structures and algorithms used to realize the Tbox and the Abox are specified (one could easily imagine implementing either in a semantic net-style language). Many of the issues that have occupied our time previously, while important in their own right, were not really addressing KL-ONE functionality, but more how to implement certain things. For example, "classification" is a way to implement the capability of answering the question "does description x subsume description y?" by essentially computing the answer in advance, and then placing the description in such a way that the answer can be read off directly when needed (by simply looking for a "SuperC" chain between descriptions). "Inheritance" is another SI-Net issue, since the ability to know which properties a description has is independent of whether they are "stored locally" or "inherited". Further, any talk of links at all, or of "local Roles", impinges on SI-Net concerns, and not on the power of the KL-ONE language.

SI-Net level issues have had a great deal of discussion in the past; we now want to try to address ourselves to the functionality of the KL-ONE language, independent of SI-Nets. This means, for instance, understanding that answering questions about subsumption and providing compositional terms for the Abox to use in making statements are the primary goals of the Tbox. The new goal of this discussion, then, was to understand the functionality of the Abox, without worrying about SI-Net issues. We wanted to begin to characterize the competence of an assertional component without committing ourselves to a particular syntax or a particular implementation style.

Terminological Issues

As it turned out, our presentation of the relations among the Terminological Component, the Assertional Component, and SI-Nets raised as many issues about "terminological competence" as it did about "assertional competence". The driving force behind what we call "terminological competence" is the ability to manage the subsumption relation between Concepts. In other words, it is up to the Tbox to "know" when one Concept subsumes another. In this sense, the Tbox does indeed embody knowledge, but not world knowledge like the Abox. Much of the discussion in the first session centered around what exactly the knowledge of the Tbox amounted to.

A suggestion, for example, is that a SuperC link could be taken to have assertional force; that is, that the link between a Concept such as DOG and, say, ANIMAL might be thought of as asserting that all dogs are animals. Our view is that the SI-Net level link between the two Concepts is not intended to capture any such assertion, but rather the fact that, by stipulation, the Concept called DOG includes as part of its definition the one named ANIMAL. There cannot be any doubt as to whether all instances of the lower Concept are instances of the higher one since that is the way the lower Concept is defined. If you want to use a Concept with the name DOG in such a way that it does not necessarily carry ANIMAL with it, a Concept not subsumed by ANIMAL must be used. Given such a Concept, you would then be free to assert (using the Abox) that all of its instances are animals.

The knowledge in the Tbox is embodied in linguistic constructs that resemble noun phrases more than sentences. Rather than a sentence like "all dogs are animals and have four legs", the Tbox might contain a phrase like "an animal with four legs" somehow related to the name DOG. There will be sentential constructs in the Tbox, but used only in order to construct nominal constructs like "a person such that one of his parents is a doctor". The big question, then, is precisely what kind of "noun phrases" should the Tbox support and what should be the subsumption relationships among them.

Our feeling is that these "noun phrases" can be broken down into two major categories: compositional and primitive Concepts. A compositional Concept is one whose meaning can be completely understood in terms of the meaning of its parts and the way these are composed. For example, if QUADRUPED is the name of the Concept "an animal with four legs", then to be a QUADRUPED is precisely to be an animal and to have four legs. So when introducing the term, we might say something like "By a QUADRUPED, I mean exactly an animal with four legs". A primitive Concept, on the other hand, might be introduced by something like "By a DOG, I mean, among other things, an animal with four legs". The difference here is that it is necessary but not sufficient to be a four legged animal to be a dog, the way the term is being used. So while in the case of QUADRUPED, animal and four legs gives the complete story, in the case of DOG, it does not. In particular, there could be two distinct primitive Concepts with the same internal structure.

This leaves the question as to the what the internal structure of primitive Concepts should be. Rusty Bobrow has suggested that primitive Concepts need "local Roles" to account for the Roles introduced as the Concept is formed. On the other hand, it might be possible to put all the structural components of the primitive on a compositional Concept, then merely introduce the primitive as a subConcept below. For example, instead of requiring a "tail" Role on the DOG Concept, first consider the (compositional) Concept of "a quadruped with a tail" and then treat DOG as a primitive subConcept of it. The net effect is that a primitive is an atomic Concept except for its superConcept.

However, as David Israel points out, the primitives cannot really be primitive until this last remnant of structure (the superConcept) is removed, placing the Concept at the top of the taxonomy. Under this interpretation, primitives are connected to other Concepts only in terms of the other compositional Concepts that use them. As David argues, if the motivation for primitive Concepts is anything like natural kinds, and the competence of the Tbox is purely linguistic, then DOG should not be below ANIMAL since knowing that dogs are animals is not a matter of knowing a language.[1] However, another

[1]An alternative view of Israel's is presented in "On the Semantics of Semantic Networks", to appear in an upcoming issue of IJCM.

way of looking at this is to say that independently of how natural kinds are treated, there may be valid reasons for wanting terms which are neither compositional nor atomic. Mike Freeman observed, for example, that there are Concepts (like TENANT) that only seem to make sense in the context of other Concepts (like RENTAL-AGREEMENT).

As far as compositional Concepts are concerned, their structure consists of Roles and Structural Descriptions ("such that" clauses) which specify how the parts of the Concept come together to form a whole. Roles in KL-ONE have two aspects: a "member" aspect which determines what kind of entity can fill the Role, and a "set" aspect where the fact that a Role can have several fillers is taken into account. The member aspect of Roles is characterized in terms of a "value restriction" (or V/R) which is a function from Concepts and Roles to Concepts. For example, the V/R of ARCH and LINTEL is BRICK; the V/R of GREEN-ARCH and LINTEL is GREEN-BRICK. Note that the same Role can be used in both cases; at the Tbox level, there are no "local Roles" and "mods" links. Similarly, we can dispense with the notion of "taking V/R's conjunctively" and just use the single (compositional) Concept that is the conjunction.

Our feeling about "local Roles" is that there is a less implementational view of the intuition behind them. What they are intended to represent is the fact that each Concept has its own unique version of a functional role that is inherited from some superior Concept. So, for example, we can have a FOOTBALL-WIDOW be "a spouse of a football fan" or talk about "a child of a doctor". But "child of a doctor" is not a different role than "child of a person" - it is just that there are two slightly different Concepts of the fillers of the same functional role in two different Concepts. (Note that one can consider the problem one of confusing the SI-Net object-type, "Role", with real, honest-to-goodness, functional roles - what Roles were intended to represent.) This is what we take "QUA-Concepts" to have been intended to represent (in particular, we take "QUA" to be a function from Concepts and Roles into Concepts). While the V/R of a Role in a Concept determines what type of thing the filler of the Role can be, the QUA-Concept of a Role in a Concept is defined as the thing that fills the Role, whatever it may be. One thing that distinguishes QUA-Concepts from Roles themselves is that Roles also have a set aspect, which we now examine.

Currently, the set aspect of Roles is characterized in terms of a "number restriction" (or #/R) which is a function from Concepts and Roles to integer intervals. The interval is intended to bound the size of the set of the individual role fillers that are associated with an instance of the Concept. For example, the #/R of MAMMAL and LEG might be <2,4> while that of QUADRUPED and LEG is <4,4>. Again, the same Role can be used in both cases.

Without even worrying about properties of sets other than cardinality, a number of design decisions arise with respect to number restrictions. One way of looking at a #/R is as a description of the size of a set. This immediately suggests more general forms of descriptions such as "an animal with an even number of legs" or "an animal with either 2 or more than 6 legs". Taking this to the extreme, we might define BIPED as "an animal with as many legs as there are prime numbers between 14 and 22". If the Tbox is supposed to know that BIPED is subsumed by "an animal with either 2 or 4 legs", we are assuming a lot more competence than the ability to notice that one interval falls within another. In the general case, the subsumption question becomes undecidable.

One possibility proposed by Danny Bobrow is to treat the above BIPED Concept as quite different from "an animal with 2 legs" even though we, with our knowledge of number theory, know that they describe the same class of animals. The point is that, as number restrictions, "2" and "the number of prime numbers between 14 and 22" are incomparable since they are structurally quite different. So, the question as to whether one number restriction is stronger than another reduces to a recursive question as to whether a number Concept is subsumed by another and can use all the same Tbox machinery as before. For this to be well-defined, however, there has to be some level of #/R that does not require a recursive analysis of number Concepts. At this level, the Tbox uses numbers (for example, the standard min-max interval) without conceptualizing them.

The notion of #/R raises some interesting problems regarding the nature of subsumption itself. The BIPED example above suggests that subsumption is based only on the structure of the Concepts and not on their meaning. What then can and cannot be concluded about meaning from subsumption? For example, if we have Concepts standing for sentences and one subsumes the other, can anything be said about the relationship between their truth values? As Mike Freeman observed, the subsumption lattice seems to be related to a lattice of implication but not in any immediate or obvious way. (Our feeling here is that the implication relationships between sentences are part of the domain of the Abox, not the Tbox.)

Again related to the #/R issue, when do we want to say that a Concept like "an X such that P" subsumes "an X such that Q" for some Structural Descriptions P and Q? One possible answer is that the former subsumes the latter when the sentence P subsumes the sentence Q (given a structural criterion for subsumption over sentences); another answer might be whenever Q logically implies P; still another, might be whenever the Tbox can conclude P from Q (under some resource limitations).

Given that subsumption partially orders the set of Concepts, the only way two Concepts can be mutually subsuming is when they are the same Concept. So, for example, if "an X such that P" and "an X such that Q" are mutually subsuming, then, in fact, there is only one Concept here, the canonical form of both of these. As Bill Woods notes, this may be a problem when the notion of subsumption is powerful enough since it may be undecidable how to represent a Concept described by "an X such that P" since this involves locating its canonical form.

So far we have shown how "mods" links can be understood in terms of the V/R and #/R functions which take both Concepts and Roles as arguments. Differentiation, on the other hand, is what is used to actually introduce new Roles. As with Concepts, we envision two kinds of Roles, compositional and primitive, introduced by two kinds of differentiation. Compositional differentiation defines a new Role by restricting the V/R of an existing one. For example, a MALE CHILD of a person is a CHILD that is male. This Role is compositional since the condition is both necessary and sufficient. Primitive Roles, on the other hand, are really "new" in that no sufficiency conditions are present. For example, the PRESIDENT of a company might be introduced as a special OFFICER of the company. Just as we can imagine all Concepts as being subConcepts of some primitive Concept THING, we can imagine all Roles having a primitive Role PART as their ancestor.

This completes the discussion of the components of the Tbox. It does, unfortunately, leave a number of issues unresolved. Very little has been said about the nature of Structural Descriptions. To a certain extent, these depend on an analysis of propositions for the Abox. Because Structural Descriptions accentuate the interdependence of Roles, we will probably also want to have a method of introducing large conglomerates of Roles and Concepts in an incremental (and maybe mutually recursive) fashion. The naming problems here start to become quite significant. In fact, the whole issue of names has not been addressed at all. Our feeling is that names do not belong in the Tbox since they are handles that refer to Concepts or Roles in the Tbox. However, primitive Concepts and Roles are somewhat like lists in LISP: because there can be two of them with identical structure, the only way to be sure you are getting the same one repeatedly is to name it. Compositional components, on the other hand, need never be named since they can always be referred to unambiguously by their structure, much like atoms in LISP. At any rate, the part of KL-ONE that deals with names does not seem to fit comfortably in either the Tbox, Abox or at the SI-Net level.

Desiderata for a KL-ONE Assertional Component

The Assertional Component (Abox) is concerned not with concepts (terms), but with sentences. It uses terms from the Tbox to say things about the world - to make statements that can actually be believed by the system (the Tbox itself makes no commitments to belief).

We think of the world as composed of individuals and relations among them. We want the Abox to be a possibly incomplete model of the world (and perhaps other possible worlds). This has two important consequences:

- o the structure of the Abox i.e., knowledge of the world, does not necessarily match the structure of the world itself; it just has to say things about the world;
- o we need a way to make weak statements in the Abox, for example,
  - . statements that do not mention any particular individuals (e.g., existential and universal statements)
  - . disjunctions,
  - . and negations.

KL-ONE has had a crude Abox all along, comprising Nexuses and Contexts. Unfortunately, this framework forces us to make statements about particular individuals. In other words, while the current KL-ONE mechanism may be sufficient for making certain strong statements about the world, it will often force us into having too much information. We propose dropping the Nexus/Context mechanism altogether, in favor of a language with emphasis on weak statements. This is a prescription, it seems, for a language like that of First-Order Predicate Logic (FOPL). However, our motivation here is the ability to make weak statements. The language of FOPL is not necessarily the right answer (it has its own problems; in particular all predicates have equal status); however, some language of this type is needed.

Finally, we must consider propositions as conceptual entities. It is necessary to be able to consider a proposition without asserting it as holding. Thus it appears that the conceptual content of sentences to be asserted must be available in the Terminological component, freed from any assertional consequences. If there were structures for propositions in the Tbox, it would appear that the Abox would be quite simple in structure: all propositions to be asserted would be formed in the Tbox, and the Abox would simply contain assignments of truth values to those propositions ("true" assigned to a proposition would constitute belief in that proposition).

Assertional Issues

One of the first issues that this divergence from the Nexus/Context "party line" raises is the addition of a substantially different type of entity to the Tbox. We must distinguish between propositions - "proper objects of belief", as David Israel puts it - and terms, which cannot be asserted. Rusty Bobrow would prefer to see a scheme with more of the flavor of the old-style Nexuses, wherein "object-centered" statements are given a prominent place. In particular, propositions would be handled just as Concepts currently are. It is clear that a more FOPL-like language could handle the existence and coreferentiality statements that Nexuses and Description Wires specialized in; moreover, it might be argued that any notion of a special representation giving prominence to object-centered statements is an SI-Net implementation consideration.

The proposal to handle propositions as Concepts (and not as syntactically distinct Tbox objects) demands an accounting of the meaning of connecting propositional Concepts to Nexuses. One suggestion is that we consider the interpretation of such Concepts to be as "holdings", so that to assert a proposition one would connect the Concept for it to a Nexus, thereby stating that a "holding" of the proposition "exists". Once again, the notion of a proper object of belief comes in - can one maintain this similarity without forcing "things" (objects, etc.) to be objects of belief? It is generally agreed that the objects of belief are proposition-like and not thing-like (one can believe "John is tall", but not "John"). Whether or not the syntax of proposition-like things and object-like things is different, all agree that any reasoning mechanism using them must be able to tell the difference. It seems straightforward to manifest this distinction in the syntax of the Tbox, although one could imagine a reasoning mechanism knowing that subConcepts of the Concept PROPOSITION were to be treated differently from all other Concepts. It is quite possible that propositions are important enough in an "epistemological" sense to justify providing additional syntax for them.

One issue in the debate over whether Propositions constitute a syntactic type distinct from Concepts is the meaning of certain taxonomic relations when composing them. Is the proper interpretation of multiple superConcepts for Proposition-Concepts the logical AND of the parent Proposition-Concepts? By the same token, is the interpretation of multiple subConcepts of Proposition-Concepts their logical OR? The notions of description composition (as in e.g., STUDENT&TAXIDRIVER) and logical conjunction (IT-IS-RAINING & IT-RAINED-YESTERDAY) are very similar. One component of the community would like to see the same mechanism used to handle both. If there is a notion of combining propositions to make compositional propositions that is different from simple conjunction of truth values, then the proposition-as-Concept idea won't work. Perhaps the two could share an implementation mechanism, but that should be independent of their real conceptual similarities and differences.

Another question that arises when considering propositional objects is whether or not there is a fixed set of proposition-forming operators. For a given logic, one fixes the set of special logical symbols in advance. Is this necessary here? Should there be an easy way to extend the set of logical operators? One proposal is that KL-ONE come equipped with a set of primitive Concepts that represent a basic set of proposition-forming operators (like it does for LISP datatypes). Since these would be Concepts, one could easily extend the set to include new operators.

David Israel suggests that perhaps a single proposition-forming operator

- "APPLY" - would be sufficient. APPLY could use Concepts like AND to construct propositions. It was said that there is some psychological evidence that the set of proposition-forming operators develops in people over time; perhaps this is some indication that Concepts like AND and OR are the right way to pursue this.

One final question has been addressed here: what place do reasoners play in a knowledge representation system like KL-ONE? The classifier is provided as a part of the knowledge representation system, since it implements a logical capability that has been determined to be part of the kernel of KL-ONE (i.e., description subsumption based on structure of Conceptual "terms"). Are there other types of reasoning that should be provided in a similar fashion? One faction of the community holds that paying attention to reasoners, per se, is misleading. They are generally conceived of as things that make use of a representation over a particular domain, but do not have anything to say about what we want to be able to express. On the other hand, the respondents feel that propositions, for example, "get their life" only from the reasoners that use them. One point of confusion here is the incomparability of the classifier with say, a natural language reasoning system: as stated above, the classifier is an implementation of a certain functionality, one that gives the representation language part of its meaning. Reasoning mechanisms that deal with domains bear no such special relationship to the language.

1.2 The Role of the Realizer in KL-ONE

Chaired by William Mark (ISI)

[Note: Bill Mark began this session with a seminar on "Realization", a process that attributes new descriptions to individuals as a system learns about them (see Mark's paper on page 78). The following report assumes that the reader is familiar with that material.]

The appropriate role of realization in KL-ONE depends on how much of its activity is (1) necessary according to the definition of KL-ONE, (2) useful as support for _any_ reasoner using KL-ONE, and (3) useful as support for specific reasoners using KL-ONE (e.g., Consul). These issues must be explored for both realization in general and for the limited form of realization described in Mark's "Realization" paper.

The first issue can be explored in analogy with classification: if assertion is as much a part of the KL-ONE language as definition by composition, then realization is as necessary to the formalism as classification. Put another way, "if there is any meaning for assertions, there is an aspect of realization that goes part and parcel with assertion and not with reasoning" (R. Brachman). That is, just as the classifier maintains--in fact _defines_--the intensional knowledge structure of KL-ONE, some part of the system must maintain (and thus define) the extensional structure. Otherwise, assertion will have no enforced meaning in the language. For example, if KL-ONE had a mechanism for stating and asserting propositions, and if P(x) and Q(x) were asserted, the assertion of the conjunction P(x)&Q(x) would presumably be considered part of the language, and would presumably be part of the realization task. The argument here is that if the language allows assertions, it must provide and enforce a semantics for them--otherwise they have no explicit meaning in the language, and are therefore not really part of it.

This argument applies to KL-ONE as a formalism with well-defined semantics for definition and assertion; it is somewhat hampered in the current KL-ONE by the lack of a fully developed definition scheme, and especially by the glaring lack of a thoroughgoing assertional mechanism. The argument is further muddied by the necessity of limiting both classification and realization in order to achieve computational feasibility. That is, even when we know what is right, we cannot always do it, nor (in some cases) do we ever expect to be able to.

For example, the classifier is supposed to be responsible for all intensional reasoning and the realizer for all extensional reasoning.

However, it may well be the case that some forms of necessary intensional reasoning are too expensive to perform every time they are possible, but are practical if called on only to support necessary extensional reasoning. This occurs in the current classification/realization scheme with respect to common subconcept creation. All common subconcepts should be explicitly represented in the network, since they are "always there" intensionally. Lacking this, the classifier should always know about them implicitly, perhaps creating them whenever necessary to support a specific classification. Unfortunately, in Consul we find both of these methods impractical. Common subconcept creation therefore occurs only as required by realization, thus blurring the boundary between intensional and extensional representation requirements.

In the world of limited realization and limited classification with respect to a limited formalism, the issue of necessity to language definition is ill formed. The question becomes whether it is better to make limited realization part of the language definition or to allow (force) each user of the language to provide his own mechanism (or perhaps omit it, thus possibly changing the language definition). I would argue that it is much better to provide the incomplete mechanism as a standard--to reduce the front-end burden of using the language, to maintain commonality of the language definition across applications, and to ensure that the necessary processing considerations appropriately influence the language design. Moreover, I would make the stronger statement that the Consul realizer represents a necessary first step for any realization scheme. I think that all of what it does must be done in any KL-ONE based system that uses the RSR and nexus features of the language.

There is still a problem with that, though--a problem of efficiency: does the realizer do enough good things to justify its use? The Consul realizer certainly provides the realizations that Consul needs; it also provides many realizations that Consul never uses. In the case of Consul, this "inefficiency" is totally justifiable because (1) we have no way of knowing which realizations we will end up using, and therefore have no criteria for eliminating some; and (2) we definitely need the information produced by the realizer, and know of no other way to obtain it. But in other systems--especially those able and willing to use more domain-specific shortcuts--it may be possible to do all of the necessary extensional reasoning without resorting to realization. This would undoubtedly be a more efficient mechanism for doing this kind of reasoning, but it would also undoubtedly be less general, less principled, etc. The position taken in Consul is that the realizer is the only reasonably general mechanism for providing the information we need, and that it is not so costly that it cannot be used. Finer judgement on this issue must await more experience.

All of the discussion so far applies only to the first issue: realizer actions required by the definition of KL-ONE--insofar as these can be determined for the current KL-ONE. There is one area, the escape mechanism of executing procedures attached to paraindividuals, where the Consul realizer goes beyond this. This is certainly support for extrinsic reasoning rather than support for the formalism itself. I think it applies to outside reasoners in general: it is a "hook" for any kind of reasoner to do processing outside of the formalism. It is the only such support facility in the current realizer, and I think it is a necessary one (see the next section).

With regard to the issue of how much of the Consul realizer is specific to the reasoning processes of Consul, I would say none at all. I could not argue that the whole realizer might not have been designed differently for a different application, but there is no specific part of the existing program that has any special connection to Consul.

Extensions

Possible extensions to the realizer fall into several categories:

- extending use of propositions;
- taking into account information about real world "environments" and groups of nexuses;
- dealing with a larger class of differences when comparing an incoming description with descriptions already known to describe certain nexuses;
- broadening the class of extensional reasoning done in the realizer;
- dealing with beliefs.

KL-ONE must certainly extend its ability to represent and assert propositions. As mentioned earlier, there are several implementation proposals for this in the making (see the summary of the technical discussion on assertions, Section 1.1, page 8). When either (or both) of these proposals are incorporated into the language, the realizer must be enhanced to take general propositions into account. The difficulty of providing this facility depends on what the propositions are about: if they refer to a single nexus, handling them is well within the existing operational framework; if not, the realizer will have to have new ways for finding the information that is relevant to a particular realization.

For example, as Hector Levesque points out, we must be able to represent statements about partial knowledge, such as "there are no white cows in the field" or "messages from Smith are usually short enough to display on my screen". Systems must also deal with statements about real world environments, configurations of objects, etc. For the realizer, the difficulty is how to use this knowledge once it has been asserted. Propositions like the one about messages from Smith must influence other realizations and must ultimately influence externally observable system actions.

This adds a *new* dimension to realization. The realizer will have to consider real world entities and descriptions that are only indirectly related to incoming real world entities and descriptions. For example, realization with respect to some context will have to take into account propositions about that context, and perhaps other objects in that context (if we know that one of the three people in the room is the murderer and decide which two people in the room are not murderers, then we know who can be described as the murderer, etc.)

Realization by checking RSR constraints is a well contained problem. Extending realization to take into account the assertion of propositions, environments, etc., will make realization more complex, but the mechanisms needed are fairly well-understood and their incorporation is foreseeable. The next step, presumably something to do with the use of non-propositional descriptive information, is a bit more difficult to formulate. The realizer presumably should do something when the "realization set" (the almost sub- and superconcepts of the incoming description) contains differences involving presence of additional roles, more restrictive VR's, etc. The definition of a clean mechanism for doing this is not obvious, at least at present.

No matter what is suggested for incorporating wider classes of propositional and non-propositional descriptions into the realization scheme, I would suggest that there will always be a need for "hooks" that connect extrinsic reasoning processes to realization. Even in the unlikely event that we figure out how to represent everything we want in KL-ONE, there will always be a need to shortcut reasoning processes based on KL-ONE alone in order to achieve efficiency, make use of existing reasoners or special hardware, etc. The problem is to keep these hooks appropriately isolated, to keep the extrinsic knowledge highly visible, and to allow it to be used only in a constrained manner (e.g., I do not think we would ever want an extrinsic reasoner to be able to alter network definitions).

Another possible area for extension of the realizer is the incorporation of additonal inferential capabilities. One example of an inferential capability that is clearly related to realization but is not handled by the current *realizer is deciding* that two entities previously thought to be distinct are in fact the same--and then implementing that decision by merging the relevant nexuses (or something). Another example suggested by Rusty Bobrow would use network relationships to decide that "if A exists, then so does B" (e.g., if A is a See Event and B is a Message that fills its object role). Still another example suggested by Danny Bobrow is inference on the basis of knowledge about sets: if a SeeAllMessagesInMailbox Event had been applied to a Mailbox, then you could conclude that a Message in that Mailbox had been seen without an explicit representation of that fact. Doubtless there are many more examples.

I can think of no principled reason not to move the realizer in the direction of broadened extensional reasoning. The above examples of inferential capability are not fundamentally different from the kind of reasoning that currently goes on in the realizer--they are just elaborations. I see the realizer continually growing as long as interested parties add such capabilities to the current scheme. However, this will make it increasingly difficult to maintain a separation between realization and application-dependent reasoning.

Finally, an interesting area of future research with direct bearing on realization is the representation of belief. In many systems it is useful to model all descriptions of the real world in terms of belief. This gives rise to the problem of representing multiple beliefs about a single real world entity. Attaching descriptions to real world entities must take into account the belief framework in which the description occurred. There must be rules for assuming belief in one framework into belief in another, rules about when beliefs conflict, etc. Much of this processing is beyond the realm of realization (it must involve domain-dependent reasoning). However, the realization scheme must provide an adequate foundation in which belief-oriented reasoning can take place.

1.3 Individuality in KL-ONE

Chaired by William Mark (USC/ISI)

Introduction

Despite the fact that individual concepts (IC's) are a well defined part of the KL-ONE formalism [1], there has been (and, to some extent, continues to be) debate about the representational functionality of individuality in KL-ONE: the meaning of IC's, the possible need for additional mechanisms to express facts about individuality, relationships to formal logic, etc. This paper discusses some of the facets of this debate, some of the decisions that have been made, and some of the issues that remain unresolved.

The Need For Individual Descriptions

Everyone agrees on the need for some representation of individuality in KL-ONE. In order to make certain deductions about objects, it is necessary for a reasoner to know whether a description can apply to only one object in the world or to a variety of objects in the world. For example, "uniqueness of reference" is required in order to decide whether an object can be described as "filling the role" of a description of another object. Figure 1 shows a KL-ONE description involving several concepts and The specified description of nexus N1 could be paraphrased as "seeing a message from Jones"; the description of N2 is "a message from Jones". Now, under what circumstances can we say that the object represented by N2 has been "seen", i.e., has filled the object role of a See Event? Certainly not when all we know is the generic description of N2: N2 is merely described as "Message'-ish"; the use of that description as the VR of the object role of See Event' is part of the definition of See Event', not a statement about N2. In order to say that the entity represented by N2 is the object of the entity represented by N1 (described as See Event'), it is necessary that Message' refer to at most one nexus. If Message' could not possibly describe any other nexus besides N2, N2 must be <u>the</u> thing that is the object of See Event'. This sort of deduction is certainly necessary for realization (see Mark's paper on "Realization", page 78) and presumably for other reasoning processes as well.

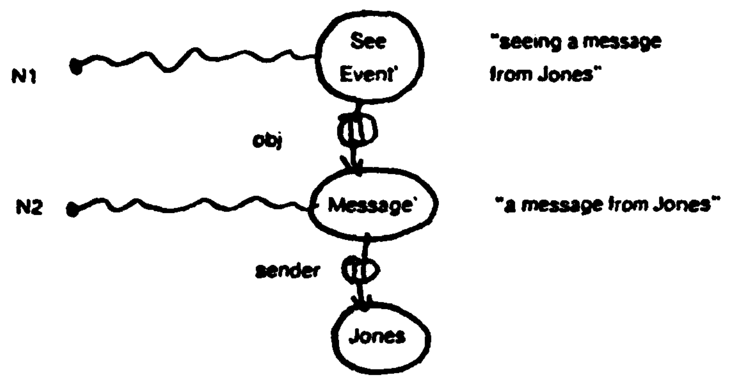


FIG. 1. THE MEANING OF DESCRIPTIONS IN KL-ONE

Problems With KL-ONE IC's

This "uniqueness of reference" property is certainly the purpose of KL-ONE IC's: an IC is intended to be a description that can be wired to at most one distinct nexus in any context. Unfortunately, IC's have two shortcomings that prevent their use for this purpose in the Consul system:

- o the assertion of descriptional individuality which is carried by an IC is not relative to a specific context or contexts;
- o IC's cannot be further specialized.

It is often useful to specify descriptions that are known to "refer uniquely" under some (extensional) circumstances, but not others. This is not part of the KL-ONE IC mechanism; IC's are meant to be individual in every context. Also, there seems to be no intrinsic reason why individuality should be related to some property of ultimate specialization: the system might discover a more specific version of an individual description. These two points are interrelated in that when individuality is relative to a context it might be useful to preserve the more general IC in a "base context" while further refining it to be uniquely descriptive in later contexts (see the section "Individuality and the KL-ONE Formalism", page 28).

The issue of further specialization can be addressed by choosing to simply copy an existing IC and then further expand on it, rather than using inheritance and further specialization of IC's. This could in fact be considered more appropriate if there is no reason that IC's should be able to subsume other IC's for the sake of proper taxonomy or classification (this is a matter of continuing debate).

Given the difficulty of specifying individuality with respect to a context or contexts, and the (at least) ambiguity raised by the further specialization problem, I proposed a different scheme for handling individuality in the Consul system--one that totally eschews KL-ONE IC's.

Individuating Concepts

In Consul, all descriptions are via KL-ONE Generic Concepts. Some of these have inheritable attached data called "ISpecs". An ISpec on a description associates sets of contexts with sets of constraints. The constraints associated with a context or contexts in an ISpec are equivalent to an appeal to an external process to determine whether the description describes uniquely in the corresponding context(s). When any reasoning process (e.g., the realizer--see page 78) wants to determine whether the description refers uniquely in a context, it checks to see if the contextual constraints specified in the ISpec are true of the context and the structural constraints are true of the description.

Currently, the structural constraints[1] used in Consul ISpecs are of two types:

- o a set of roles of the candidate description whose VR's must refer uniquely in the given context;
- o an additional description which must be satisfied by the denotation of the candidate nexus in the given context.

In addition, numbers and strings are always individual descriptions.

The basic process for determining whether a description is uniquely descriptive in some context runs as follows:

[1]The contextual constraints are not worth discussing--the only type allowed now is a list of context names. In order for the constraint to be satisfied, the name of the context of interest must be on the list.

1. If the description is already wired to more than one mutually distinct nexus in the candidate context, then either decide to merge the nexuses or trivially conclude that the description is not uniquely descriptive;

2. If the ISpec specifies a set of roles, and each role has a VR description that is uniquely descriptive in the candidate context, then conclude that the description is uniquely descriptive (this is basically a structural recursion that terminates with strings or numbers);

3. If the ISpec specifies an additional description and the additional description is satisfied by the denotation of the described nexus in the candidate context, conclude that the description is uniquely descriptive;

4. Otherwise, conclude that the description is not uniquely descriptive.

There seems to be general agreement that this computable, context-relative individuality is necessary. However, there are some problems with the current scheme (mostly having to do with "additional description" constraints), and some inadequacies inherent in the overall approach.

The intent of an additional "individuating" description, say **Message''**, is to allow the statement of constraints like "if the nexus described as **Message'**, say, a message from Jones, can also be be described as **Message''**, a message with a unique message number in some unique individual's mailbox, then **Message'** must describe uniquely. Unfortunately, this descriptive mechanism does not do the job: no deductions based solely on the external entities being described can suffice to prove the uniqueness of some internal description. Such deductions must take into account the description itself and the taxonomy in which it is embedded. The "additional description" constraint in this example only serves to guarantee that a nexus is distinct; the uniqueness of some description of that nexus is a different question.

For these constraints to make any sense at all, they must apply to descriptions rather than to the external entities being described. That is, the constraint would have to be a metadescription that would have to be satisfied by some <u>description</u> as a test for descriptional uniqueness. Unfortunately, it is not clear how to do this in the current KL-ONE formalism. This leaves us in the situation that many (though not all) believe that the individuation formalism must have some constraint specification mechanism beyond the "sets of roles" form--but no one knows how to do it. It is a problem that merits further research.

An inherent characteristic of the ISpec scheme is that it can only make extrinsic statements (for the benefit of an interested extrinsic reasoner) about the uniqueness of reference of particular descriptions; it says nothing about the terminological status of individual descriptions. This is generally viewed as a good thing--the problem of making such determinations for external reasoners is seen as separate from the "terminological competence" problem of deciding what individual concepts are, what their function in the KL-ONE formalism is, etc. That is, although (once cleaned up) this scheme of using ISpecs to define "individuating concepts" is useful and probably necessary for reasoning in KL-ONE, it does not solve the problems that require terminologically individual descriptions.

Individual Terms

Consider the following representation problem (suggested by Richard Fikes): represent in KL-ONE the concept "person with hometown T", where T is some (individual) hometown. The scheme presented in the previous section cannot be used to do this; it allows a given description to be seen as an individual, not to be arbitrarily created as one. We need a mechanism for expressing arbitrary individual terms--not to use in the kind of reasoning discussed in the previous section, but to use to form other descriptions. It is thus analogous to the application of the epsilon term-forming operator of first order predicate logic <u>epsilon</u> xF(x), meaning "<u>some</u> x such that F(x)".

This is a job for KL-ONE IC's. A "starred" IC, i.e., one <u>arbitrarily</u> designated as taxonomically distinct, can be formed to represent any desired individual term (see Figure 2). Note that we know nothing more about the description T than that it is "some individual town"[2].

A question remains as to whether KL-ONE needs "non-starred" IC's. These would be individual terms formed not arbitrarily (i.e., according to extrinsic principles), but compositionally (according to the normal principles of KL-ONE taxonomy). Their use would thus be analogous to the iota operator of first order predicate logic, <u>iota</u> xF[x], "<u>the</u> x such that F(x)".

There is general agreement that this is in fact an appropriate interpretation of non-starred IC's in KL-ONE. The question is whether they are necessary given the ISpec scheme and starred IC's. A useful point of view (again generally agreed upon) is that the successful checking of ISpecs on some description as discussed in the previous section <u>allows</u> the system to form an IC representing that description as an individual term. It must be a

[2]There is some debate about whether or not the starred IC is merely the terminological equivalent of a nexus.

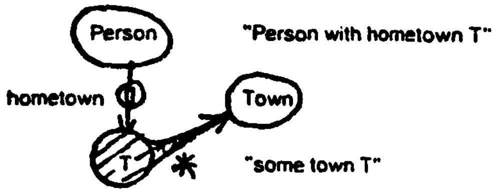


FIG. 2. USING STARRED IC'S

direct instantiation, that is, no new information (other than its individuality) can be added. The issue of whether IRoles (or indeed any roles on IC's) are necessary is still open. Also, there seems to be no solid viewpoint on the proper use of the IC once it has been formed, given that its origin may depend on extensional factors (e.g., contextual constraints) that are not directly encoded in its definition as a term. I think that a good summary of the general viewpoint would be that compositional IC's are legal terms, but that their appropriate use by the system is unclear. There seems to be no general sentiment that they are absolutely necessary to the formalism.

Individuality and the KL-ONE Formalism

There remain a number of "larger issues" concerning the epistemological and semantic status of individuality in KL-ONE. For this purpose, it is useful to separate three notions of individuality:

1. descriptional uniqueness independent of context;

2. descriptional uniqueness dependent on context but independent of context state;

3. de facto descriptional uniqueness in the current state of a given context.

The first type corresponds to the current KL-ONE IC's: if an object fitting such a description exists, then there is at most one such object in any context. Hector Levesque points out that such concepts have the property that their description of a distinct nexus in some context is evidence that there are and can be no other nexuses which they describe in that context.

Rusty Bobrow provides the following analysis of the second type of individuals: if a generic concept is viewed as a predicate G(x), the question of individuality is whether there is one and only one object in the context of interest that satisfies G. If we assume that a context is a monotonically growing set of assertions describing a world, then what we mean by type two individuality is that we can prove by those assertions in that context that if there exists an x such that G(x), then in that context, there is exactly one x such that G(x). The problem is: what assertions can we use for this purpose? What form must such assertions have? Does a single description wire constitute such an assertion? If contexts are non-monotonic, then the existence of a single description wire is an assertion of a very particular sort: it cannot be taken to mean that there can be no more than one such link, but it should be taken to mean that at least one object exists which satisfies the predicate implied by the description attached to the wire. On the other hand, the assertion formed by wiring two descriptions to a single nexus might be enough to prove the individuality of a single description.

To Tom Lipkis' objection that this results in individuality that is true or false depending on the current state of the context (i.e., that these are really type three individuals), Rusty points out that it is really a matter of individuality being true, false, or unknown. Since the model described by the context is incomplete and growing, it is not always possible to deduce whether any given description is individual. What we are left with is the change from not knowing to knowing (whether or not some description is individual) as further assertions are added to the context. If we can prove that some description is individual (or is not individual) then by monotonicity it will remain individual (or not individual.)

This allows the interesting reading that it is not the individuality of a description that changes as assumptions are added to a context, but only the system's knowledge of individuality. For example, at one time the system may be unable to prove a description G is individual and may also be unable to determine that G is not individual. Later on (as more assertions are added) you may be able to determine that it is indeed an individual (or not).

David Israel notices that this entire discussion addresses epistemology and not semantics. That is, it addresses what is "known", "not known", and how all of that is figured out. Consider the following example of a problem in the representation of terminological individuality: suppose we have a concept translating roughly as "a natural satellite of the Earth"; suppose a particular context represents the actual world; suppose we say that, in that actuality, there is one and only one natural satellite of the Earth. The extension of the predicate in the actual world context has membership 1, but in some other context it might have membership 73. Regardless of that, it is still true in every other context that, in the actual world, the extension is a singleton. That is, we need to be able to form a rigidifier: a term for "<u>the</u> natural satellite of the Earth". The concept "a natural satellite of the

Earth" will never be, semantically, an individual, regardless of what properties we paint onto it that reflect whatever assertions we make about the actual world (or any other context.)

Another interesting point for speculation (also noticed by David Israel) is the status of ISpecs as part of the KL-ONE language or outside of it. In particular, should they be viewed as theories expressed in the language? Put another way, is an ISpec part of the structure of the concept (like a role) or is it a comment within a certain context about that concept (a theory). It is not a question of whether or not it <u>can</u> be part of the structure, only of whether it <u>should</u> be (David thinks not). For example, suppose there is a notion of a California Town; it has a role for "name". Now it is possible to imagine two different theories about California Towns: one that says that no two California towns have the same name, and one that says that some California towns can share names. Do we need two different theories for the <u>same</u> concept California Town, or <u>two different</u> concepts? David maintains that we should be able to entertain alternate theories about the same concepts. According to one theory, name is an individuator, and according to the other, it is not.

Mike Freeman points out that this is related to his notion of "QUA" (see Section 1.8, page 55). Concepts like Town get names from some sort of context in which you give towns names. Knowing where certain roles originate may give you leverage on whether or not they individuate. It may thus be possible to distinguish "names" by virtue of fundamental intensional reasoning (e.g., perhaps they are the result of different naming events).

Conclusion

I think that a fair statement of the status of individuality in KL-ONE is that, ignoring the occasional lacunae, we have a reasonable grasp of its functionality in the formalism. Individual descriptions can be formed and used in a consistent manner. As usual, the deeper meaning of this process as a way of representing knowledge in the machine is less well in hand.

REFERENCES

[1] Ronald Brachman (1978)
A Structural Paradigm for Representing Knowledge. Bolt Beranek and Newman Inc.

1.4 KL-ONE System Maintenance

Chaired by Jim Schmolze (BBN)

Issues Addressed

KL-ONE has grown in two important senses over the past few years, namely in terms of its size (and maturity) as a representation language and in terms of its popularity as a usable system. Users of the INTERLISP implementation have had to deal with an experimental system that contains its fair share of bugs and whose definition lags the theoretical work on KL-ONE by over a year. Additionally, the past year has seen the transportation of KL-ONE to the Jericho and Dolphin machines, with a Franz Lisp version for VAX machines to be completed in the near future. All of this has been in parallel to the development and usage of a Smalltalk KL-ONE implementation, known as KloneTalk, developed at Xerox PARC.

All the problems of maintaining a unique theoretical KL-ONE language with incarnations in various programming languages and for various machines is far too large to be addressed in this session. Rather, methods for maintaining and improving the BBN INTERLISP version will be the primary focus with a look at the possibility of sharing code with the KLoneTalk implementation as a secondary topic. The problem of how to decide which features of the theoretical KL-ONE language should be incorporated into the BBN implementation, although an important topic, will only be addressed if we finish discussion of the other topics, as this question will require a near infinite amount of discussion.

Proposal by Jim Schmolze

The BBN implementation is currently the source of all instantiations of KL-ONE, with the exception of KloneTalk, so I suggest that we limit the maintenance portion of this discussion to the BBN system. My proposal is the obvious one, which is to continue operations in the same manner as they now function. BBN receives all bug reports, requests for modifications, suggested extensions, etc. and BBN maintains the software. Among certain selected users, BBN will release sources and encourage assistance for software work with respect to both fixes and extensions. All such software items must go through BBN to become integrated into the system. The organization of multi-site software work will require some protocols for non-BBN programmers to avoid overlapping work. These protocols for non-BBN programmers are simply

1. Before undertaking substantive work, submit a note to BBN describing what is to be done. Then wait for a reply. BBN will take care that the task is not already under way and that the programmer has the latest sources. This will be necessary because non-BBN sites will NOT be advised of all software modifications as they are happening.

2. Small modifications can be done as needed by non-BBN sites; in fact this is encouraged. Simply advise BBN of the modification either while it is in progress or after it is done. However, if a programmer does not perform step (1), no guarantees can be made that a conflict will not arise.

3. When complete, submit the modifications to BBN in the form of "patch" files which contain precisely those functions and other entities that have changed. Please do not send new versions of entire KL-ONE files as that can make integration much more difficult.

As to sharing code with KloneTalk, I have no concrete suggestions. Has anyone else?

The question of how we decide what software work to do is difficult, however, there is an easy default. Namely, we all discuss our wants but whoever actually does the programming gets his way. This would apply for BBN and non-BBN people alike. Of course, this strategy works only when we deal with packaged extensions to KL-ONE. When changing underlying KL-ONE structures or algorithms, BBN must perform the work unless special arrangements are made. For such modifications, BBN must be convinced to undertake the effort.

I must also say that I am not happy with my proposal for this part, but I do feel that it is a realistic suggestion. It might be good for us to consider forming a KL-ONE software steering committee. However, the primary problems are person-power limitations and the fact that if KL-ONE is to change radically, we should consider re-implementation along with modification, which would require a great amount of development time.

Summary of the discussion: BBN Implementation of KL-ONE

Our session began with a brief discussion about maintenance procedures for the BBN INTERLISP implementation of KL-ONE. We reviewed the chairman's suggestions for inter-site maintenance, as described in the preceding section, and agreed to adopt them.

We briefly entertained the idea of sharing either ideas or software between KloneTalk and INTERLISP KL-ONE and quickly determined that sharing ideas was certainly possible whereas software held no such hope.

Implementation Efforts

Richard Fikes began by offering some interesting news. KloneTalk is currently running up against the limits of the Smalltalk it uses. He must decide between upgrading to a new version of Smalltalk, which would require a substantial effort, or using an INTERLISP version of KL-ONE. One possibility is to use the BBN version. However, Richard expressed strong regret at leaving the user interface of KloneTalk behind. In any case, his and others' interest in a new INTERLISP KL-ONE were discussed.

There was a fair amount of excitement at this prospect, which began by leveling all possible criticisms against BBN's KL-ONE:

- It is large and awkward, making it difficult to modify in major ways.
- It does not have a uniform design.
- There is duplication of effort in the software.
- For TOPS-20 users without extended addressing, KL-ONE is simply too large to use for any sizeable project.

We also tried to analyze the reasons for the difficulties in BBN's KL-ONE software and came up with two reasons. First, the design of KL-ONE evolved as experience was acquired, causing earlier designs to become outdated. Without enormous person-power, it was impossible to keep the implementation updated in this fashion. Second, the original design of the software was thought to be clean, but only with respect to a certain set of principles. However, our principles changed over time.

All in all, we recognized that a new implementation need not mean a better one. In fact, it could very well suffer from the same traps that BBN's KL-ONE has fallen into. However, there is some merit in re-implementing simply because a subset of the design is quite solid and could be re-done quite well. In conclusion, we decided that with the new functionality that has been discussed during this Workshop, a new implementation must include all of the old functionality that we are confident of, plus attempts at new components. And we have no reason to believe that we will not look back at this new implementation several years from now without regrets, at least with respect to the new components.

For now, we will add the user interface components that are most needed (discussed in the session on utilities) to BBN's KL-ONE and will attempt to design for the new functionality. When that design is clear, we will decide whether we can put it into the current KL-ONE implementation, or start afresh.

As an aside, Richard Fikes noted that his KloneTalk system did not heavily utilize the inheritance of operator definitions, which is part of Smalltalk. This was a bit of a surprise since many of us feel that an object-oriented design would be well suited for a new KL-ONE. We should examine Richard's experience in this regard carefully.

Design Documents

In addition to the conclusions already stated, we decided that solid designs for new components of KL-ONE must be done before considering implementations. This means that design documents must be done. Although none were assigned at this time, our intention to write them was recognized by all.

Integrating the Classifier

A brief discussion of integrating the ISI classifier (see Tom Lipkis' paper on page 128 and the discussion summary on page 36) into BBN's KL-ONE followed. We agreed to add several flags and options to the classifier to augment user control, but our primary decision was to have the classifier remain a function that is invoked only by the programmer. It will not be invoked automatically whenever concepts are created or changed.

1.5 Classification

Chaired by Tom Lipkis (ISI)

The session on the classifier began with an explanation of the current implementation, followed by a discussion of several issues regarding the design of a classifier and its integration into KL-ONE (see Tom Lipkis' "A KL-ONE Classifier" on page 128 which describes the KL-ONE classifier in detail). Several of the points brought up were possible improvements to the algorithm, or existing bugs. Most of these points are reflected in Lipkis' paper.

An important issue is how far the classifier should go in trying to determine subsumption. Should it have a reasoning ability that allows it to prove predicates in RSRs? Should it be able to understand complex descriptions such as "the 57th digit in the decimal expansion of pi?" (these issues are discussed in more detail in Lipkis' paper). No definite conclusions were reached about kind of reasoning the classifier should perform, but there was general agreement that some reasoning is appropriate in many cases, and that it should be possible to limit the resources that can be consumed in such processing. Such limits, as well as the incompleteness of the reasoning engine (even if given unlimited resources) imply that the actual taxonomy may only be a partial representation of the virtual subsumption relationships. That is, some concepts will have been found to be "incomparable" by the classifier, even though by expending more effort, they could be shown to have a subsumption relationship. The consensus was that resource limited classification results in taxonomies that are incomplete, but not incorrect, and that this would not lead to any false inferences if handled correctly by other processes.

Most of the integration issues were addressed during the system maintenance session, but one which was discussed in some detail concerns the action of the classifier when it discovers that two or more descriptions are identical (i.e., they describe the "same" virtual concept, even though they were created as distinct structures in the network). Ideally these structures should be merged into one, as together they have only one correct place in the taxonomy. In practice, however, this may be difficult or undesirable, as external processes may have handles on parts of the structure (in the form of names of roles or concepts, or LISP pointers to the actual structure) which may be invalidated by such merging. The classifier currently maintains a record of merged concepts, to allow processes to update their pointers, but this is not sufficient because some information is lost when the merging occurs. It was requested that the concept merging be made optional, the alternative being to leave the identical concepts as siblings, and merely inform other processes that they are actually identical. (Note that this results in an incomplete taxonomy, as discussed above.)

1.6 KL-ONE System Utilities

Chaired by Jim Schmolze (BBN)

Issues addressed

This discussion concerned software "tools" that are needed by users of KL-ONE. We chose the tools that are most needed and ordered them in terms of their priority. For some tools, we planned their initial designs and our methods for constructing them. These tools included

1. facilities for reading/printing KL-ONE networks from/to files (a KL-ONE I/O facility),[1]
2. facilities for in-core editing of KL-ONE networks,[2] and
3. facilities for viewing or browsing[3] in-core KL-ONE networks.

The discussion was primarily directed at, but not limited to, the INTERLISP implementation of KL-ONE. However, much insight was gained from the utilities that already exist in the KloneTalk[4] system.

---

[1]A KL-ONE I/O facility would allow one to write a portion of a KL-ONE network onto a text file and read it into a different KL-ONE system, creating an equivalent copy of the original portion.

[2]A KL-ONE editor would allow one to modify the KL-ONE network while it is in-core, thereby allowing one to readily view the modified concepts with all their inherited information.

[3]A KL-ONE browser would offer various types of information about a network from general network structure to the definition of specific concepts.

[4]KloneTalk is a Smalltalk version of KL-ONE written by Richard Fikes of Xerox-PARC (see Richard Fikes' paper on page 90).

The User Interface to KloneTalk

We began by viewing a videotape of the KloneTalk system, presented by Richard Fikes of Xerox-PARC.[5] The tape clearly demonstrated the KloneTalk interface which allows one to view and edit a KL-ONE database. We all agreed that the user-interface to KloneTalk would make an extremely valuable tool for the INTERLISP version of KL-ONE.

The aspects of KloneTalk's user interface became important during this discussion; important enough to warrant a brief description of the salient features of this interface.

KloneTalk provides a graphical interface that lets a user

- o examine and edit KL-ONE concepts,
- o read/write KL-ONE concepts from/to files, and
- o maintain several windows on a screen simultaneously, where each contains a working context for displaying concept descriptions, editing sessions or whatever.

It is a convenient and useful tool for the KL-ONE network builder.

There is an interesting dependency between the first two components listed above. The KloneTalk I/O facility can create a textual description of a concept that could be read into another KloneTalk system to create an equivalent concept. However, that textual description is also quite easily read by people; in fact, it can be prettyprinted for just that purpose, upon which the KloneTalk concept editor relies. The editor begins a session by obtaining a textual description of a concept, thanks to the I/O facility. Then it enters the standard Smalltalk text editor. When the user has finished editing the concept, the text is read by the I/O facility and the concept <u>replaces</u> the one being edited. Note that this methodology raises some difficulties that are discussed later.

Note also that the KloneTalk editor and I/O facilities do not rely upon graphics at all, although they utilize graphics quite nicely. The only required tool is a text editor.

---

[5]This tape was also shown to the larger Workshop audience during Richard Fikes' talk.

The Primary Topics Discussed

The chairman began the discussion by listing the major desiderata, at least so far as the INTERLISP implementation of KL-ONE was concerned. These were for

- enhanced input/output (I/O) facilities for KL-ONE networks and
- in-core editors (and browsers) for KL-ONE networks.

Additional issues to be discussed were

- the status of the classifier (described in Tom Lipkis' paper on page 128) developed at ISI and the manner in which it will be integrated into KL-ONE,
- the question of the centrality of either graphics facilities, personal machines or INTERLISP functionality for development of future utilities.

Graphics, Personal Machines and INTERLISP

A consensus quickly emerged that nothing essential should be made to depend on the availability of either graphics or personal machines, unless absolutely necessary. The only example currently under consideration which would require graphics is a browser for KL-ONE networks. As to reliance upon INTERLISP functionality, we will refrain from using INTERLISP specific packages as much as possible because of the FRANZLISP translation effort.

We all expressed interest in having a browser for KL-ONE networks, and all shared the opinion that a program that could draw KL-ONE pictures might provide the best tool for browsing. Some discussion of the layout problem for KL-ONE pictures ensued, with the tacit agreement that it is indeed a difficult problem, too difficult to address at this meeting. However, all agreed that even a modest effort in this regard would be valuable. For example, it might be relatively easy to construct global pictures of a network automatically, where such pictures would show only concept names and subsumption relationships between concepts. Unfortunately, no volunteers stepped forward for this task, although the BBN contingent expressed interest in examining the problem.

An Input/Output Facility for KL-ONE Networks

The ISI contingent reported upon a package they built which reads CKLONE-like descriptions of KL-ONE networks and creates the appropriate structures. CKLONE is a KLONEUSERS package, written by Frank Zdybel at BBN, which extends CLISP to include forms for creating KL-ONE structures. The CKLONE language has three advantages.

1. It can represent a fairly large subset of KL-ONE structures.
2. It can be prettyprinted by Lisp.
3. People can easily read it.

The ISI package is similar to CKLONE but it creates KL-ONE structure when it is reading forms, as opposed to when it is evaluating forms.

The BBN group reported upon an old I/O facility called BRACKETREADPRINT. The language that this package uses to store networks is highly specialized and quite difficult for people to read. However, the output portion could be modified to generate CKLONE-like notation.

This discussion led to the following agreements:

- o Rusty Bobrow offered to modify the output component of OLDBRACKETREADPRINT to generate forms which can be read by the ISI input package.[6]
- o Tom Lipkis, Richard Fikes and Rusty Bobrow agreed to design a single language for storing networks on files that both the INTERLISP KL-ONE and KloneTalk could read. Presumably, this language will be the union of the existing KloneTalk and ISI languages.
- o Jim Schmolze made a contingent offer to construct a concept editor similar to the KloneTalk editor, where the contingency depends upon the availability of manpower.

However, some difficult problems remain with the I/O facilities in that this design will not help with the following:

[6]This package, called "NT", is now complete. It reads and writes KL-ONE networks using either ISI or Klonetalk notations.

- o merging networks,
- o maintaining syntactic validity across files (and to handle, e.g. forward references to as yet undefined names)
- o handling the problems raised by the heavy reliance on names, especially in the context of merging nets.

The Classifier

The discussion of I/O facilities also raised questions about the place of the classifier (see Tom Lipkis' paper on page 128). The ISI input facility uses the classifier as a matter of course, which several of us felt was a bad idea. After all, the classifier can be an expensive device to use and there are times when the KL-ONE programmer knows that a piece of network being loaded need not be classified (e.g., if the network being loaded has already been classified at some earlier time with respect to the current network). After some discussion, a consensus arose supporting the idea that the classifier should be invoked only under the call of the programmer. It should be available to re-classify a concept (or subnet) that was edited, or classify a new concept (or subnet) that was being integrated into an existing network. But it should not be thought of as a black box sitting between either file I/O and the KL-ONE network nor the editor and the network.

A KL-ONE In-Core Editor

Finally, the problem of in-core editors was raised, causing lively discussion about structure editors versus text editors, concept editors versus network editors, editors we could build today versus research for future editors, JARGON versus standard editors, etc. Luckily, from amongst these orations arose a modest proposal, modelled after the KloneTalk editor. We agreed that the KL-ONE editor we build next would have the following traits:

- o It would edit concepts and not networks.
- o It would use the I/O facility discussed earlier to generate readable descriptions of concepts for editing and create the resulting KL-ONE structures after editing.
- o The editor would be a structure editor versus a text editor, since the concept descriptions to be generated are list structures.
- o It will edit in-core structures.

o It will assume that all new descriptions of concepts are both complete and local.

This design requires that the ISI input package be amended to replace structure and not simply create new structure. For now, we will determine replacement versus creation based upon the names of KL-ONE objects, for both concepts and roles. We are not happy with this, but it is a realistic short-term solution. One problem left unresolved was how to handle both local and inherited information, since we want to be able to view inherited information but not edit it. KloneTalk handles these distinctions, but in a manner that was not satisfactory to the group at large.

1.7 RSRs, role orderings and quantification

Chaired by Rusty Bobrow (BBN)

Despite our intention that KL-ONE have a clear semantics, several components essential to the expressive power of KL-ONE are currently so poorly specified that no two KL-ONE afficionados can fully agree on their intended interpretation. The collection of structures broadly referred to as SD's, RSR's, and RVM's is a notable example of this difficulty. We have attempted to use these structures to simultaneously solve several related representational problems, but the range and number of these problems has made it difficult for us to provide a clear and cohesive semantic interpretation for the SD, RSR and RVM mechanisms.

The goal of this session, broadly stated, is to answer two questions:

- o What is the intended role of the SD mechanism (including RSRs and RVMs) in KL-ONE and its components?
- o In what ways must the SD mechanism[1] be modified, extended or contracted to fulfill this intent?

In particular, this session will focus on three problems, and propose a common solution mechanism. Briefly, these problems are:

- o How do we provide at least the expressive power of standard predicate logic quantification in SDs?
- o How can SDs be used to extend the types of structures and "name spaces" for structured objects available in KL-ONE?
- o How can KL-ONE concepts containing SDs be specialized by strengthening or specializing those SDs? (What are the useful and natural ways to strengthen the constraints defined by SDs?)

The first and third problems have been raised several times at this meeting and elsewhere, but the second is likely to be less familiar. Its

[1]For simplicity, we will often refer to the collection of data structures and programs in KL-ONE that define SDs, RSRs and RVMs as the "SD mechanism."

relation to the other two problems will be explained, and its importance as a representational issue will be discussed.

To place these problems in perspective we must first ask "why have SDs been included in KL-ONE?" One characterization of SD's, clearly enunciated by Michael Freeman and widely accepted within the KL-ONE community, is that they are merely "sets of assertions (propositions, logical clauses)." This is a particularly narrow view of the SD mechanism. SDs are intended to provide the glue that holds concepts together, to define the meaning of the roles which give each concept its internal structure. Clearly one way in which they can do this is simply to make a set of assertions about the role fillers of a concept. Certain concepts, however, have a structure which cannot be defined simply in terms of facts about the fillers of their roles. Simple examples include the concepts needed to describe many of the structured objects and abstractions familiar to people in computer science and artificial intelligence, such as a linear list. There are no predicates on fillers which can be used to define the first, second, third, etc. roles in a linear list, since the same set of fillers can occur in different orders in different lists (and the same filler can occur more than once in a single list).

We will propose one possible mechanism, based on SDs, for defining such structures, after we explore the ramifications of first problem: the need for quantification in SDs.

KL-ONE, in common with other frame-like languages, provides facilities for describing structured entities such as arches. It does this, in part, by stating that such an entity is composed of a number of constituents of specified type each of which fills one or more roles in the overall structure. This is not enough information to define such a structured entity. The fundamental representation issue is that an arch is not merely a pile of blocks. An arch is a collection of blocks, two of which fill the role of upright and one of which fills the role of lintel, but it is not true that every collection of three blocks forms an arch. There are other constraints that must be met for us to say, for example, that a particular block fills the lintel role in some arch.

SDs provide the mechanism needed to specify these further constraints, to state the predicates that must hold true of the fillers of certain roles in a concept as a condition of their filling those roles. Thus, for an arch to exist, two blocks must fill the upright roles, and one block must fill the lintel role, each upright must "support" the lintel, and the two uprights cannot "touch" each other.

In KL-ONE, predicates such as "support" have traditionally been represented in terms of generic concepts, (for example, a concept called SUPPORT), in order not to expand the number of primitive categories of structure which must be supported by the KL-ONE system. Thus, when we define the generic concept ARCH, we must make use of the concept SUPPORT, but if the generic concept ARCH is to be taken as a "schema" for defining the structure of individual descriptions of an arch, then the use of the SUPPORT concept must be restricted or parameterized in terms of the roles of the ARCH concept. In particular, we must be able to specify the structure of particular instances of SUPPORT as a function of the constituents of each individual instance of an ARCH. Individual instances of ARCHes and SUPPORTs will be represented by KL-ONE individual concepts or ICs, and we use KL-ONE ParaIndividual concepts (PIs) to specify the desired functional relation between the fillers of roles in an individual ARCH and the argument roles in the related SUPPORT ICs. This is indicated in Figure 1.

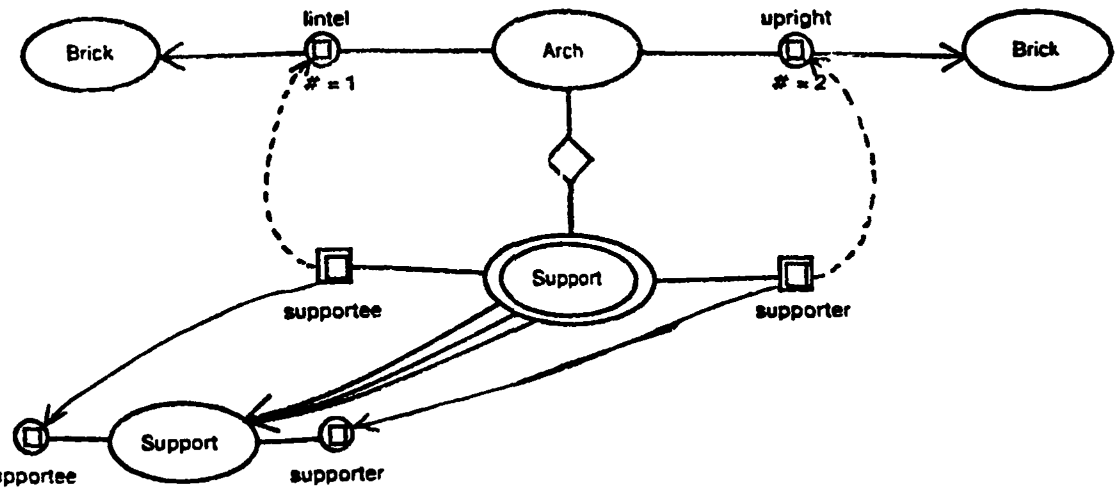


FIG. 1. THE GENERIC CONCEPT ARCH WITH A PARAINDIVIDUAL SUPPORT CONCEPT

Note, however, that this specification of the support relations for an ARCH is ambiguous. Do both uprights jointly support the lintel? Does each upright individually support the lintel? Does only one upright support the lintel, and if so which one? The ambiguity here arises because we have RoleSets in KL-ONE, specifying a set of similar roles (e.g. the two lintel roles in an ARCH), and we have not specified a mechanism for defining a set of predicate instances in terms of one or more sets of roles. We have provided the equivalent of lambda expressions, in the form of paraindividuals, but we have not specified how to provide the expressive power given by the quantifiers in standard logical formalisms.

To reiterate, while paraindividuals provide a way of specifying a single predicate or related object[2] determined by the roles of an Individual concept, they do not, by themselves provide a way of specifying sets of such predicates or objects. Thus, given the standard KL-ONE description of an ARCH, which has a Roleset upright with cardinality 2, we must distinguish among the following possibilities:

- there is one SUPPORT relation with two supporters and one supportee, or
- there is one SUPPORT relation with a single supporter (just one of the uprights!) and supportee, or
- there are two SUPPORT relations, each involving a different one of the uprights of the ARCH.

The notion of a Role Set Relation or RSR was introduced as a way of meeting this need. A PI can be viewed as a schema for IC's, with a set of holes or arguments to be filled in. Each element of the set of predicates/objects described by an SD is the result of plugging in the fillers of some tuple[3] of IRoles in the Individual concept into a corresponding set of roles in a PI. Given this view, a essential component of an SD is a means of (partially or completely) specifying a set of tuples of IRoles. Such a specification is exactly what is done by the quantifiers in a prenex formula in a "typed" predicate calculus.

[2]I will use the phrase "predicate or object" to abbreviate "predicate or object or anything else describable by a KL-ONE concept", since we do not have a single name for this class of entities.

[3]We must have something like ordered n-tuples, rather than unstructured sets of IRoles, since we must specify the way in which the IRoles in the collection are to correspond to the arguments in the PI.

At this point two questions arise:

- o Should we be quantifying over (or forming relations among) the roles of a concept or over the fillers of the roles?

- o What are the desirable characteristics of a language for specifying relations or sets of tuples? (What is lost by simply using standard typed quantifiers, for example?)

The answer to the first question depends in part on an issue to be discussed later, that is, the representation of structured items such as linear lists and plans. A partial answer can be given without touching that issue, however. The entities being quantified over must be distinct, otherwise quantification makes little sense. But one of the important features of KL-ONE is that one entity may fill several roles in the same concept. Thus, a single person might be fill the two roles _vice-president_ and _comptroller_ in a single company. We would really like to quantify over the fillers as they participate in the framework established by the concept as a whole, i.e. "the fillers as they (each) fill some role." At the level of ICs, this notion is captured by the KL-ONE construct of an IRole, an individual role, and we view the proper domains for quantification to be the sets of IRoles defined by RoleSets.

As to desirable characteristics, two critical ones are some form of completeness and perspicuity. One touchstone for completeness is the ability to express all relations expressible as quantifier collars in some typed predicate calculus. Such quantifier collars define relations by characterizing sets of assignments for named variables. These assignments are constrained in two ways: first, by giving the domains over which each variable is to range, and second by giving a set of _combinatorial_ constraints such as whether all the elements of some domain must be represented in the set of assignments (this is part of the meaning of a universal quantifier). Since the class of relationships expressible by such collars depends on the types of domains that can be specified as type restrictions, a subsidiary question we might pose is "what sets of roles do we want our quantifiers to range over?"

This point engendered some discussion during the technical discussions. It was generally agreed that we wish to range over the RoleSets which form part of the concept to which the RSR is attached. In addition, all of roles which could be defined by rolechains should be available as domains for quantification. Thus, for example, in Figure 2 we could have as domains the _arms_ of a PERSON, the _hands_ of (any of the) _arms_, and the _fingers_ of (any of the) _hands_ of (any of the) _arms_ of a PERSON.

There was some disagreement as to whether certain constraints should be expressed as restrictions on the domains of quantification or as part of the

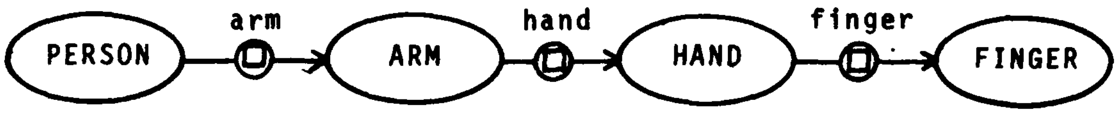


FIG. 2. ROLESETS DEFINABLE BY ROLECHAINS

predicate being quantified over. For example, suppose one wanted to say that the fingers of a person (i.e. the fingers of the hands of a person) are all connected to the hands in which they play a role. In predicate calculus terms we could express this fact by either of two logically equivalent expressions[4]:

```
Ah/hand(Person),Af/finger.of.hand(Person):[(f in
  finger(h)) => connect(f h)

Ah/hand(Person),Af/finger(h): connect(f h)
```

It seems that the second expression is more perspicuous, because the

---

[4]The quantificational notation used here is one in which the quantificational prefix consists of typed quantifiers separated by commas separated from the matrix expression being quantified by a colon; the typing of the variables is indicated by a slash followed by the domain; and the letters A and E are used to indicate universal and existential quantification, respectively.

first expression includes in the matrix the predicate "f in finger(h)". That predicate seems more naturally a part of the specification of what objects the "connect" predicate is being asserted of. Thus, the participants in the technical discussions suggested that the ability to specify such "dependent domains" be made part of the language for domain specification for quantification.

Several issues were discussed with regard to the perspicuity of the representation language. In particular, we want to compactly express facts about the relationship between sets that are somewhat awkward to express with the ordinary quantificational apparatus of the predicate calculus. If for example, one wants to say that a relationship R is a 1:1 correspondence between two sets A and B, then one must say:

```
Ax/A,Ey/B:xRy &
Ax/A,Ay/A,Az/B:[(xRz&yRz)->x=y] &
Ax/A,Ay/B,Az/B:[(xRy&xRz)->y=z] &
Ay/B,Ex/A:xRy
```

where the four successive clauses of this conjunction deal respectively with specifying that every element in the domain A has an image in B under the relation R, R is a function, R is one-to-one, and R is onto B.

In the course of the above specification, the domains A and B are mentioned 5 times and the relation R is mentioned 6 times. If the specification of either of these domains or of R is complicated, then the overall expression can be enormous. Moreover, this repetitive mention of the relation and the domains hides the essence of what's going on such that if expressions such as these are given to a theorem prover or mechanical inference system, it is not readily apparent that a relationship of 1:1 correspondence is being described. Additionally, there is no obvious way in which the notion of a 1:1 correspondence from A to B can be thought of as a more specific relation than a 1:1 mapping from A into B, since there is no single constituent of the above specification that can be thought of as the specification of the relation. (If a compact notation for typed quantifiers is not used, the situation is even worse.)

Similarly, if one wants to express some of the facts that are compactly expressed in English with the locution "and vice versa", then one must redundantly mention the matrix sentence and the domains of quantification in question. (E.g., "Every little boy loves a puppy and vice versa").

Thus our goal is to devise a notation in which relationships between sets are compactly represented without redundantly mentioning either the domains or the matrix expression being quantified. In addition, the "quantificational fact" asserted about that expression and those domains should be distinguished as a single constituent that is capable of being analyzed and taxonomized in a space of quantificational relationships. The ability to taxonomize such quantificational collars will be important when concepts using such quantifiers must be placed within a taxonomy of concepts.

The notation proposed here hinges on the observation that a quantificational statement is essentially an assertion about one or more domains of quantification and some matrix expression. Thus, we would like to specify it in a notation in which each of those domains and the matrix expression appear as single constituents and the remaining quantificational import appears also as a single constituent (possibly containing analyzable substructure). For example, the relationship "for every x there exists a y" in a statement meaning Ax/A,Ey/B:P(x,y) should be expressed as a single constituent of the representation--not as a fact derivable from scattered pieces of the representation. This can be done if we think of the quantificational import of such an assertion to be essentially an assertion about sets of ntuples from the cross product of the domains in question.

The particular language used to express assertions about relations or quantificational collars must still be specified, and this was left to be worked out in a design document promised by Rusty Bobrow and Bill Woods. Rusty proposed that a useful starting point might be found in the operations of the relational algebra proposed by Codd for work in relational data bases. These have the advantage that not only is the quantificational collar (the assertions about the relation) visible as an independent constituent of the structure, but the relation itself can be viewed as a single entity, and can be used as part of the description of the structure of the concept as a whole.

In any event, the language for describing quantificational collars must have the expressive power of standard quantification (this has been shown to be the case for the relational algebra), and it should allow easy inclusion of particularly useful assertions about relations as terms that can be independently defined and used. Among the quantificational operators that can be individually defined and then combined with ordinary logical connectives are:

- ONTO(x,y) - equivalent to AE(x,y)&AE(y,x)
- ONE-TO-ONE(x,y) - equivalent to the uniqueness conditions in the definition of 1:1 correspondence above
- AT-LEAST(n,x) - asserts that there are at least n elements of the domain of x for which the matrix is satisfied

- ATMOST(n,x) - asserts that there are at most n elements of the domain of x for which the matrix is satisfied
- EXACTLY(n,x) - asserts that there are exactly n elements of the domain of x for which the matrix is satisfied
- ALL(x) - asserts that the matrix is satisfied for all elements of the domain of x
- ALL-COMBINATIONS(x1,x2,...,xn) - asserts that the matrix is satisfied for all combinations of elements from the n specified domains

The formalism should also be capable of specifying a variety of subtle variable dependencies--including those that have motivated notions of "branched quantifiers" for expression facts that would be expressed in English by locutions such as "for every x there is a y and for every z there is a w such that P(x,y,z,w)."

The role of RSRs in extending the range of structures readily expressible in KL-ONE

Once we have introduced such a notion of a mechanism for defining relations among the roles of a concept (as opposed to simply predicates on the fillers of roles), this opens up the range of structures readily described in KL-ONE. To understand what this means, we must first take a look at one of the purposes of roles in KL-ONE.

KL-ONE concepts can be looked at as a way of providing "family names" for entities which are to be treated in a uniform way by some class of processes (often external to KL-ONE). For example, the concept LEGAL-PERSON might be used to group together those entities that were to be treated in a uniform way by the some body of "judicial" procedures (and might include both human beings and corporations, but not partnerships).

In an analogous sense, a role can be viewed as giving a name for a class of entities, specified relative to some other entity, which are considered to have similar properties. Thus, the _officers_ of a company might be treated in a uniform manner by procedures that deal with the responsibilities of the principal members of an organization.

Currently in KL-ONE, we have only one way of introducing a structure in the set of roles for a concept. This is the role differentiation mechanism, and it allows us to define a fixed, finite collection of RoleSets for each concept. With this mechanism, however, it is difficult or impossible to define a structure with an indefinite but distinguishable number of roles, such as a linear list. Here we would like to specify a RoleSet _item_ for the items in the list, a distinguished sub-role of item called first, and a linear ordering next among the sub-roles of item. We could then write procedures that operated on the elements of a linear list in sequence, by considering the roles first, next(first), next(next(first)), ... and in that order.

The incomplete proposal for RSRs described above would provide a means for defining such a structure on concepts. If the Role Set Relations can be used independently of the paraindividual mechanism, it would be possible to specify that a LINEAR-LIST was a concept with a RoleSet item, a RoleSet first which had one element and which was a differentiation of item, and an RSR next which was a linear ordering on the RoleSet item. It should also be possible to define such structures as arrays, in which there are two inter-defined orderings, the notion of row-next and of column-next.

This idea was briefly discussed during the technical discussions, and a more complete specification was left to the design document on RSRs which Rusty Bobrow and Bill Woods volunteered to write.

Other issues touched upon

The following issues were mentioned but not discussed in depth during the technical discussions:

- o When a Concept is specialized, in what way can the SDs on the Concept be strengthened (cf. Mike Freeman's position paper on a calculus of structural descriptions)? Clearly, one could strengthen the individual assertions specified by paraindividuals, by replacing them with more paraindividuals of more specific generic concepts. In addition, one could strengthen the "quantifier" of the SD, as in going from "Ax/X,Ey/Y:Pxy" to "Ey/Y,Ax/X:Pxy". This could be done simply strengthening the predicate on RoleSetRelations that forms part of the SD.

- o Can SDs be differentiated just as RoleSets, to define sub-classes of constraints that form part of the definition of a concept, and to allow these sub-classes to be strengthened independently? We can readily conceive of RSR's which are sub-relations of a given RSR, in just the same manner as the differentiation of a Roleset specifies a subset of that Roleset. Since RSR's are defined in terms of specific Rolesets as domains, and each such Roleset can be differentiated, one natural differentiation of an RSR is obtained by restricting it to subsets of each of its domains given by appropriate Roleset differentiations.

- o How can both the RSR and the associated PI's be raised to the status of full structural elements of KL-ONE? By this we mean, for example, how do we use RSR's just as we use Rolesets, to specify sets of roles. Given a Roleset R linearly ordered by an RSR O, we would like to specify such subsets of R as "the initial element of R (under O)", "the tail of R (that is, all elements of R following the initial element of R)", etc.

- What is the position of RVM's relative to all this? They are rather peculiar, in that rather than dealing with roles at all, they are defined in terms of the set of fillers of a set of roles, not in terms of the set of roles at all. Clearly we need such a bridge between the world of roles and the world of fillers (which have sometimes been erroneously equated to the "intensional" and "extensional" aspects of the notation), but are RVM's the way?

- Do we want to continue using KL-ONE concepts to stand for predicates? Or do we want to introduce a new set of "assertional", "propositional" or "constraint" structures into KL-ONE?

1.8 The QUA link

Chaired by Michael Freeman (Burroughs)

As initially conceived, the QUA link was proposed as a new kind of inheritance cable pointing from the concept being defined to the RoleSet of another concept taken as its definitional context. From a purely syntactic point of view, the QUA-defined concept was considered subC to the V/R of the RoleSet it pointed to. In addition, however, it acquired as derived roles certain other RoleSets of the parent concept providing the definitional context: in particular, any RoleSet related via an SD to the one the QUA-defined concept pointed to in the parent concept was also incorporated into the QUA concept itself. A QUA link was thus a kind of multiple inheritance cable. Given the concept of leasing or renting an apartment, for example, the concept of LANDLORD would be QUA-linked to the PARTY/A RoleSet, making it subC to LEGAL/ENTITY (the V/R of the PARTY/A RoleSet), but now augmented to include additional RoleSets such as TENANT, LEASED/APARTMENT etc. (v. Figure 1). As long as the definitional context was a relational concept, there seemed to be little problem in bringing down its RoleSets into the concept being QUA-defined. A "child" from a "parentage" relationship, for instance, derives its "father" and "mother" RoleSets in a perfectly straight-forward way from that relationship. If one were to consider "child" as being QUA-defined off of the role concept of "mother", however, there would clearly be a problem in trying to derive its "sex" role from that of "mother". This issue is taken up in more detail in (4) below, where we address the original semantic motivation for introducing the QUA link in the first place.

In cases where the parent concept providing the definitional context is itself specifiable through multiple inheritance, one can induce a natural partitioning on the acquired RoleSets of a QUA-defined concept, corresponding to the different perspectives from which one can view it. Thus a LANDLORD can also be looked at as a RENT/RECEIVER, APARTMENT/OWNER, etc. In order to make explicit the source of these different vistas, we proposed marking the standard KL-ONE superC link with the label VIZ in all such cases (v. Figures 2 and 3). Insofar as VIZ links are intended to provide a focus mechanism for a problem solver (which would operate "outside" of the KL-ONE interpreter), they are functionally similar to "perspectives" in KRL. However, by restricting their introduction to QUA-defined concepts, we are making an additional claim as to what constitutes a conceptually well-formed context for their use. It has been objected (e.g., by Richard Fikes and Danny Bobrow) that one may want to ask for the RoleSets of a concept that were inherited from a particular superC, whether the concept was QUA or not. Clearly one needs some forcing examples to decide the issue one way or the other.

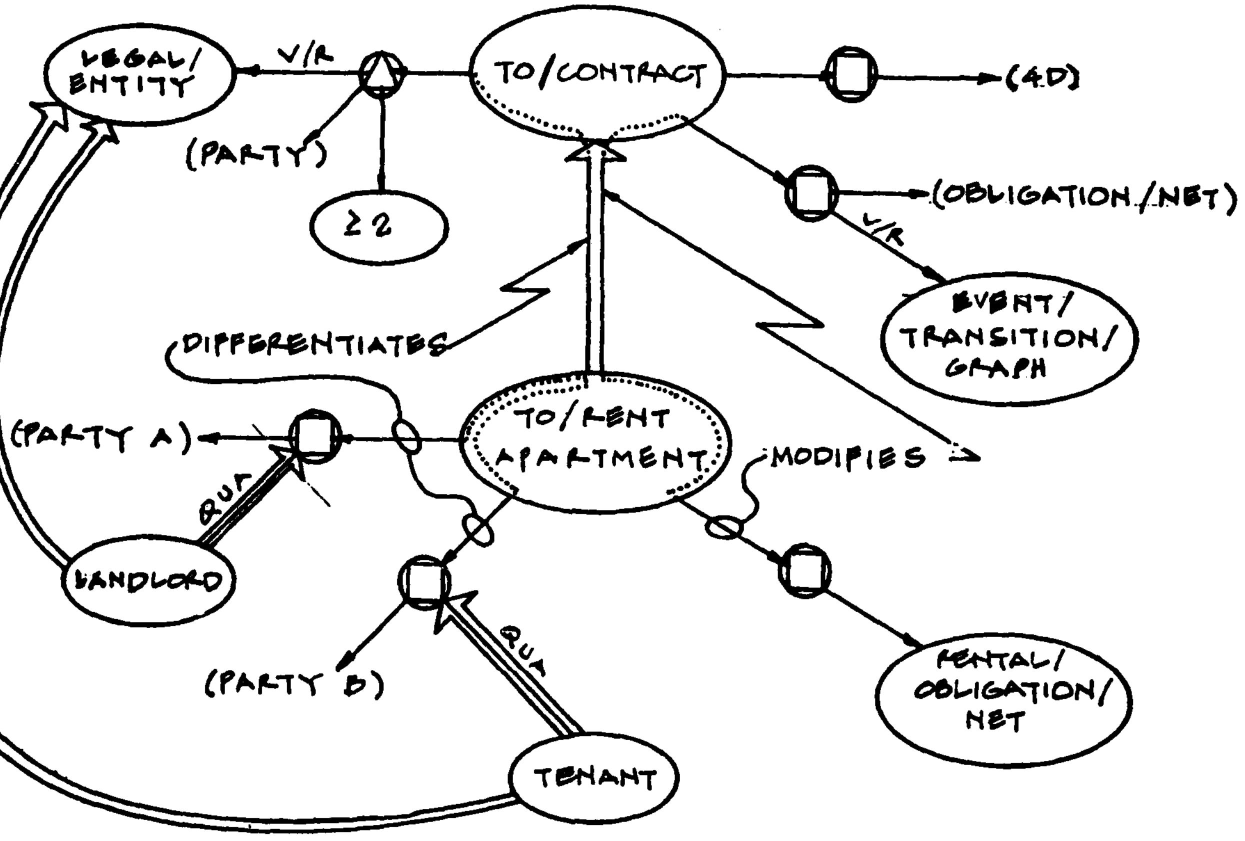


FIG. 1. QUA-DEFINED LANDLORD AND TENANT

Since it was initially proposed, the QUA link has elicited two major types of discussion within the KL-ONE community. The first concerns whether or not it can be subsumed by already existing constructs within KL-ONE. The second concerns the actual specification of the QUA link and QUA concepts.

With regard to the first, it has been suggested that the QUA link is really nothing more than a convenient abbreviation for an RSR within the so-called QUA-defined concept. In the case of LANDLORD, for instance, there would be an SD specifying that for any particular LANDLORD there must exist a corresponding APARTMENT/LEASING whose PARTY/A filler is that particular LANDLORD (v. Figure 4). Two points against this suggestion are:

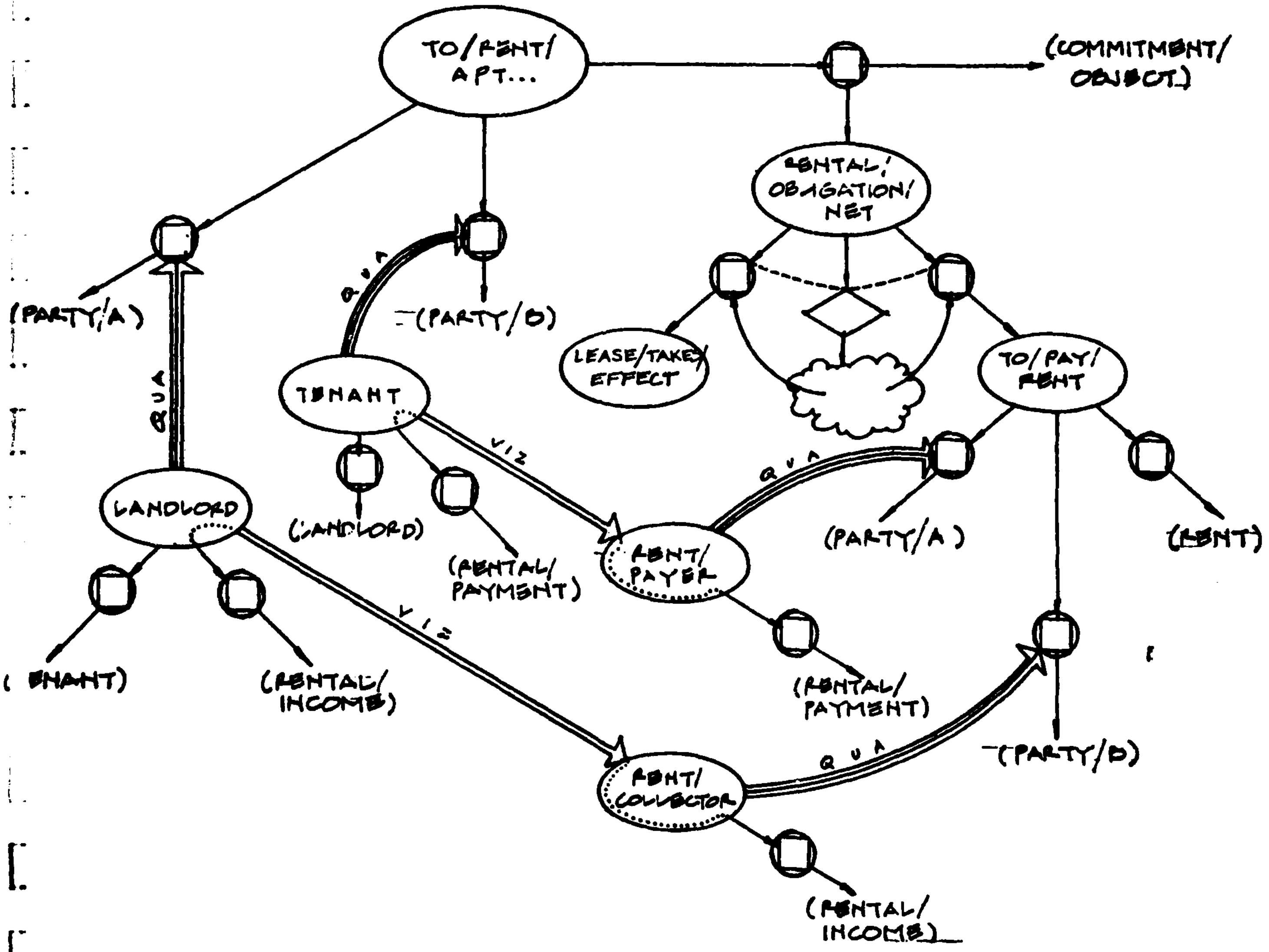


FIG. 2. THE QUA-RELATED VIZ BUNDLES

o it fails to convey the role-dependent nature of a QUA-defined QUA link

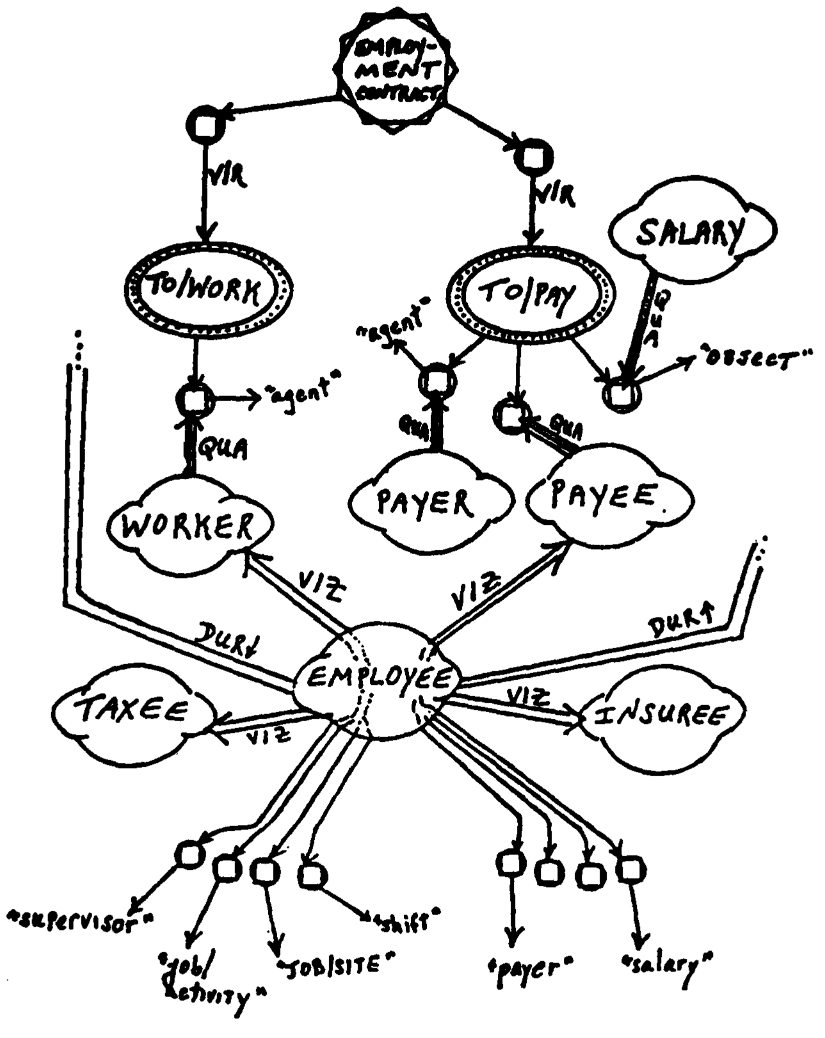


FIG. 3. THE QUA-RELATED VIZ BUNDLES

concept, e.g., a person's being a LANDLORD <u>as a result</u> of the role he plays in leasing premises he owns (a notion closer to functional composition than to aggregation or mere conjunction [cf. 4 below]);

- conversely, it seems to imply that the concept that is para-individuated (e.g., APARTMENT/LEASING) can itself be thought of

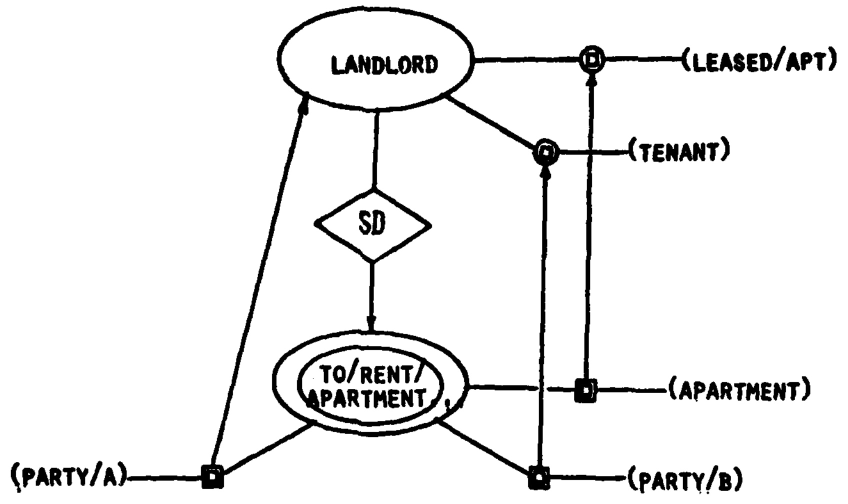


FIG. 4. RSR VIEW OF QUA CONCEPT

independently from the QUA-defined concept (LANDLORD) even though everything about the latter is totally determined through such para-individuation. This makes the SD here seem much more like what Bill Mark (following a suggestion of Rusty Bobrow) refers to as a "further description" (or FD), i.e., a circumstantial rather than definitional component (see Bill Mark's summary of the "Realization" technical discussion, on page 18, and his paper "Realization", on page 78). As a consequence, one would be forced to view QUA as an extensional rather than an intensional relationship, contrary to what was initially intended.

With regard to the actual specification of the QUA link and QUA concept, there are five main issues involved:

1) There exists another version of QUA, tentatively referred to here as RIC (for Role In Concept). As described at the second KL-ONE Workshop (see also Richard Fikes's paper on "Highlights from KloneTalk" on page 90), there is a link from the RoleSet to its RIC concept and an inverse link from the RIC concept to its corresponding RoleSet. We'll refer to this latter usage as CIR (suggesting the inverse of RIC). The concept containing the RoleSet whose RIC we're talking about will be referred to as the "parent concept".

The RIC facet of a RoleSet is directly subsumed (in the concept hierarchy) by the RoleSet's V/R facet. As such, it inherits the RoleSets which the V/R can have, but no others. (This is reminiscent of what Bill Woods suggested some time ago, namely that there should be a "para-individual" facet on RoleSets in order to represent role-dependent concepts.) If other roles are to be "derived" from the parent concept, this has to be explicitly indicated in Role Value Maps of the parent concept. This presupposes of course that one be able to establish Role-Chains over the RIC facet of a RoleSet (v. Figure 5).

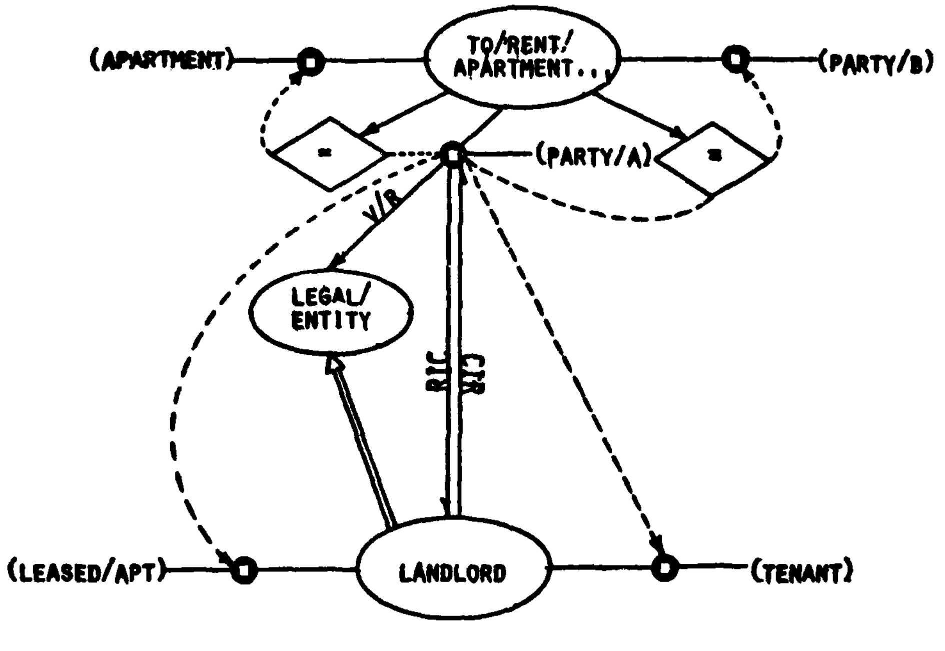


FIG. 5. ANOTHER TYPE OF QUA: RIC/CIR

It also seems that the RIC-dependent concept in such cases should have a corresponding set of RVMs with Role-Chains going over the CIR link up to the appropriate RoleSets in the parent concept. This entails however allowing one to chain from the RoleSet at the head of a CIR link to its parent concept and then back down to other RoleSets of the parent (v. Figure 6). We will refer to this as the RVM view of QUA concepts.

Insofar as the RVM view is simply a notational variant of the RSR view of QUA concepts discussed before, it is subject to the same remarks made there.

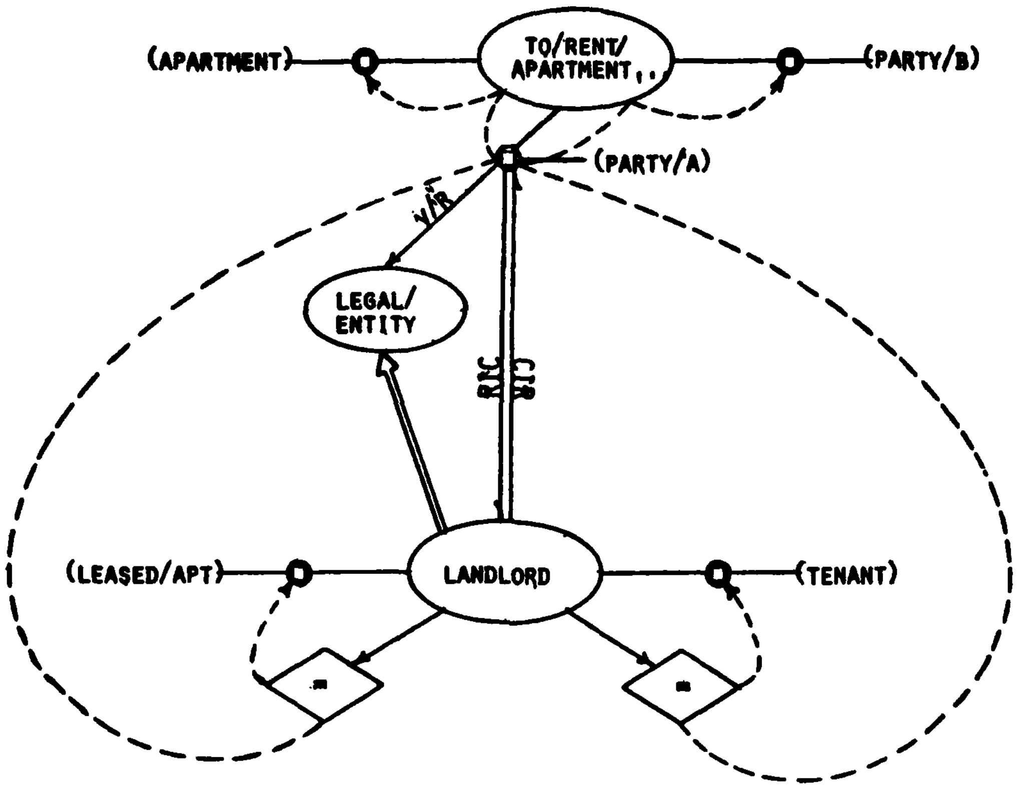


FIG. 6. RVM VIEW OF QUA CONCEPT

It was suggested that a more fruitful way of looking at the matter, however, was to make a distinction between cases in which one does or does not want to have a role-dependent concept have RoleSets automatically derived from the parent concept. Those in which they are to be automatically derived correspond to QUA, those in which they must be explicitly indicated via RVMs (or RSRs) correspond to RIC/CIR. The former can be viewed as establishing intensional role correspondences, the latter as establishing extensional ones. The fact that what is automatically derived in the case of QUA could perhaps also be represented via the RVM (or RSR) view of QUA merely obscures this point.

There was also some discussion as to whether there might be other cases than QUA in which one would want to have an intensional correspondence established between local roles (e.g., personFinger on Person being equivalent to the handFinger of the Hand of Person).

2) Given the intimate relationship between RIC- or QUA-dependent concepts and the explicitly or implicitly defining RVMs (or RSRs) associated with them, their placement on the concept subsumption hierarchy forces one to address the subsumption relationship for RVMs and RSRs. For some thoughts on the matter, see Rusty Bobrow's technical discussion summary concerning RSRs (page 44), our position paper ("Towards a calculus of structural descriptions ... " on page 115), Bill Mark's "Realization" (page 78), Bill Mark's technical discussion summary on realization (page 18), and Tom Lipkis' "A KL-ONE Classifier" (page 128). Note that currently the latter do not seem to treat RVMs or RSRs uniformly in this regard.

3) By having the QUA (or CIR) link point to the RoleSet node itself, one raises the possibility that what is being QUA-defined is the RoleSet as a whole rather than members of the set. Where the cardinality is > 1, however, one would like to have both concepts QUA-definable (e.g., BOARD/OF/DIRECTORS, as well as BOARD/MEMBER). Only in the latter case, however, is the QUA-defined concept subC to the V/R facet of the RoleSet, as originally proposed.

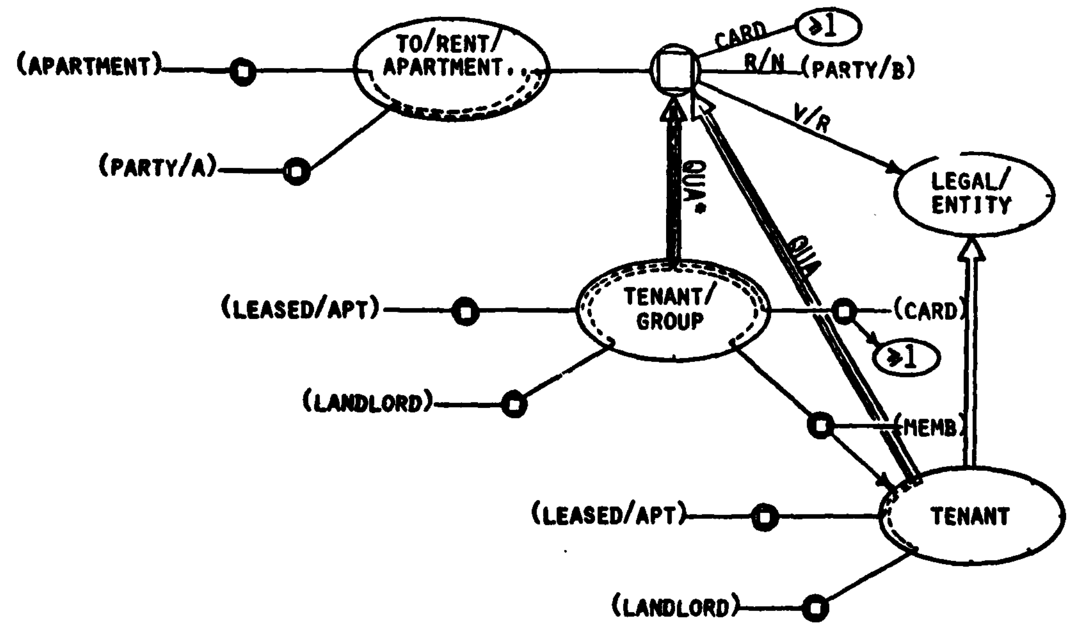


FIG. 7. TWO DIFFERENT QUA LINKS

One way of dealing with the problem is simply to introduce a second type of QUA link which treats the set-based facets or properties of the RoleSet as RoleSets themselves to be inherited by the QUA-defined concept. In place of the V/R facet, however, would be a "member" RoleSet whose V/R was the other type of QUA-linked concept that we have discussed up to now (of course, this schema would require a calculus of sets). Automatically derived RoleSets from the parent concept would in both cases follow the principles that were elaborated earlier. This type of approach is illustrated in Figure 7.

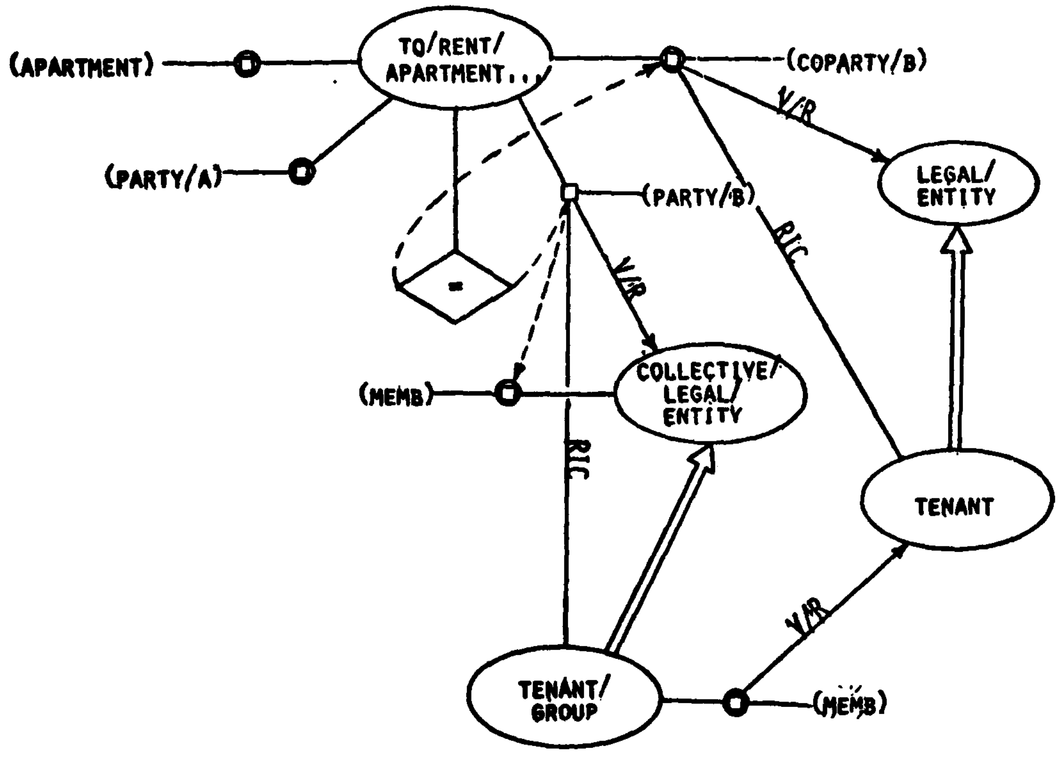


FIG. 8. RIC-LINKED COLLECTIVE VS. RIC-LINKED SET MEMBER

Another possibility is to have an additional role on the parent concept whose V/R is itself a set or collective entity, along with an RVM establishing the necessary correspondence between its "member" RoleSet and the appropriate RoleSet in the parent concept. A single type of QUA (or RIC/CIR) link could then be used for both the collective entity and the RoleSet member one. This type of approach is illustrated in Figure 8.

4) The initial motivation for introducing the QUA link was semantic in nature. The problem being addressed concerned ways in which concepts could be specialized. KL-ONE provided essentially three ways: modification (i.e., specialization of a RoleSet's V/R or cardinality facet), differentiation (i.e., creation of new RoleSets as subsets of some more general RoleSet in the parent concept), and multiple inheritance. Given that the most abstract concept in a KL-ONE taxonomy has a single RoleSet, whose cardinality facet is >= 1, one sees that the introduction of new RoleSets is in a sense ultimately always based on differentiation of that initial RoleSet. The question we asked was whether a framework could be established which would enable one to make explicit the definitional context presupposed by many, if not all, such differentiations. In the context of human society, for instance, certain types of social/contractual interactions result in the association of specific attributes with the participants of such interactions. Since these attributes derive from the role-playing activity of the participants over extended time horizons (cf. our discussion of LANDLORD and Figures 1, 2 and 3), we initially viewed the QUA link as being restricted to RoleSets in durative or iterative processes. Thus HITTER could be QUA-defined only in a dispositional sense, not a punctual one. And RoleSets of object concepts such as ARCH, PERSON, BOOK, MESSAGE were all excluded as the head of a QUA link. Note, however, that since these latter concepts can themselves all be QUA-defined relative to an appropriate framework, their RoleSets thereby also can. A MESSAGE's REQUIRED/FIELDS, for instance, have exactly the same context of definition as the MESSAGE itself, just as a LANDLORD's TENANT has. We thus hypothesized that QUA links to RoleSets other than those in durative or iterative processes can always be treated simply as a notational abbreviation along the lines just mentioned.

Note that under this hypothesis one automatically eliminates the possibility of deriving inappropriate RoleSets (e.g., "sex" of "child" from "sex" of "mother") or of creating conflicting inheritance paths for the same QUA-defined concept. The formal properties of QUA of course can be divorced from the issue of whether or not the hypothesis is correct. In the RIC framework the question in a sense doesn't even arise, since all RoleSets, from a purely syntactic point of view, have a RIC facet (just as they do a RoleName facet). Whether or not it turns out to be a "useful" representation mechanism in any particular case is not regarded as having anything to do with its implicit presence: that's simply part of the "extra-terminological" advice one might wish to make available to a user.

5) If one accepts the dependence of QUA links on process concepts, the proper representation of the latter within KL-ONE becomes crucial. We have advocated using Augmented Event Transition Networks (AETNs) as the basis for modeling the role-dependent aspect of QUA-defined concepts. An AETN is a kind of event grammar, specifying the "expected" behavior of participant role-players in processes that have extended time horizons, as well as the possible transformations that physical entities can undergo relative to particular

functional perspectives, without destroying their "objecthood". As formal objects, AETNs can be represented by sets of propositions or clauses incorporating special operators and connectives for dealing with such notions as temporal realization, precedence, possible or eventual succession, etc. These ideas are developed more fully in our position paper for this Workshop, "Towards a calculus of structural descriptions ... " (see page 115).

2. GROUP DISCUSSIONS

During the second day of the main conference, we formed several working groups in order to promote informal, technical discussions. The topics discussed were chosen from suggestions made by the participants prior to the Workshop. There were four of these groups, three of which are summarized in this chapter. The fourth group, not represented in this chapter, worked on "practice" problems using KL-ONE.

The chairperson of each group was chosen in advance and asked to prepare a description of the issues to be addressed at their session. These descriptions were circulated among the Workshop's participants at the beginning of the Workshop and participants then chose the sessions they wished to attend. This chapter presents summaries of these sessions, each written by the corresponding chairperson of each group.

2.1 Transporting KL-ONE to other LISPS

Chaired by Tim Finin (Penn)

This section discusses the results of the group meeting concerned with using KL-ONE in environments other than Interlisp. In particular we discussed Lisp versions of KL-ONE which are derived from the Interlisp KL-ONE rather than entirely separate implementations based on the same functionality. We perceive a strong interest in having such versions of KL-ONE available on machines which do not support Interlisp.

Our group explored various ways to share resources and help one another. As a whole, the group shared the feeling that we would benefit from some sort of organization, however loose, that would prevent any duplication of effort. The discussions grouped around four major issues: the nature of our interest in non-Interlisp KL-ONE translations, the nature of BBN's role, maintenance of communication and the sharing of the Franzlator translation system and any of its translations.

I. Who Wants What

The largest component of the group was interested in a version of KL-ONE which would run on a VAX. This means either a FranzLisp version or waiting for the release of Vax-Interlisp. The second largest component was interested in versions of KL-ONE for Lisp Machines. There was scattered interest in versions for several less-common Lisp dialects (UTLisp and various Lisps for Univac machines). The future possibility for a CommonLisp version was brought up and discussed.

There was a general agreement that it would be highly desirable to have non-Interlisp versions "track" the current Interlisp one. This requires mechanical translation or Interlisp emulation. The Franzlator translation system appeared to meet the needs of the task.

II. BBN's position on non-Interlisp KL-ONE

As of yet, BBN has not expressed a clear official position on its relationship to external KL-ONE users, especially with regards to those of us wishing to develop and use non-Interlisp versions of KL-ONE. The informal position has been quite supportive, both in terms of encouragement and in terms of computer resources and manpower. We recognize, of course, that the resources that can be devoted to spreading KL-ONE are very limited. The following points were suggested:

- o BBN should develop an official position on the distribution of the KL-ONE source code.
- o BBN should decide what resources and how much of them can be devoted to non-Interlisp users. In particular, it was suggested that BBN might make the DWIMified KL-ONE source code available from the Arpa network. Another particular issue is to what degree BBN will be willing to make source-level changes in the Interlisp version to simplify its conversion to other Lisps. This has already happened to a limited degree in conjunction with the effort to produce a FranzLisp version.
- o A community of non-Interlisp KL-ONE users on the Arpa net will need access to one of the BBN machines in order to send and retrieve files. We should seek some kind of support, probably from ARPA, for a computer account for this purpose.

III. Maintaining Communication

One of the more important, and perhaps simple, tasks is to set up the means of maintaining communications between non-Interlisp KL-ONE users and potential users. Three suggestions were:

1. Set up an Arpa network mailing list for those people with access to the Arpa network.
2. Set up a U.S. Mail mailing list for those people who do not have access to the Arpa network.
3. Set up lines of communication across the CSNet and CogNet networks when they become available.

Jim Schmolze volunteered to maintain the U.S. Mail mailing list for those people not on the Arpa network.

IV. Sharing Translators and Translations

The management of one or more non-Interlisp versions of KL-ONE is certainly a large and time-consuming task, involving the creation, distribution, documentation and maintenance of the code. Since none of us can take on such a large task, we sought ways to distribute the work. The general plan which we developed involved the following points:

- The Penn group will make available the Franzlator lisp translation system, the Interlisp to FranzLisp translation rules, and the Interlisp run-time support system.
- Translation rules for other Lisp dialects will be developed by the interested parties.
- BBN will make available the dwimified source code for KL-ONE.
- Each of us will be responsible for obtaining the Franzlator and the dwimified code and creating our own translation.

2.2 Inferences in KL-ONE

Chaired by Ron Brachman (Fairchild)

I. Introduction

Work on KL-ONE the past few years has, for the most part, concentrated on issues of structure. While it has always been implicit that the structure is to be built a certain way only to support inference over it, not much has been said explicitly about the kinds of inference that were in mind. The purpose of this discussion group was to try to articulate the kinds of inference that the current system and/or conception of KL-ONE covers, as well as those that were desirable to include in future work.

This is the invitation sent out to participants in the Workshop:

> The purpose of a representation language is to support certain kinds of inference. The topic we would like to address at this working meeting is, "what kinds of inference does KL-ONE support, and what kinds of inference SHOULD it support?"
>
> The notion of a classification lattice that KL-ONE has had for most of its life has implicit in it several kinds of inference: transitivity of the superConcept relation, parts inferences, etc. Can we enumerate these and make it clear just what their functionality should be? Without doing at least that, it is hard to explain or justify the language to others (logical types, especially). In addition, the assertional part of the language has been sparse, at best. What kind of assertional inferences should we plan on handling? Why is coreferentiality of description any more important than modus ponens? Can we somehow get away without a complete set of logical connectives and rules of inference?

II. Discussion

At the outset, we tried to draw a distinction between three provinces of inference in a representation framework: 1) those inferences that came "for free" as part of the language design (e.g., classification), 2) inferences appropriate to a knowledge representation system, but not considered part of the language design, and 3) those completely outside the scope of a representation system. After some discussion, we decided to drop the distinction between the first two of these, and settled on the question of what inferential power should come as part of a "turnkey" representation system, and what should not be considered the proper responsibility of a representation system.

The discussion proceeded smoothly, with most everyone agreeing on a not too extensive list of types of inference they would like to see included in KL-ONE:

- "inheritance", or the equivalent thereof (part of the discussion focused on how inheritance was an implementation notion, and not a logical one).
- classification of descriptions on the basis of their internal structure; this, too, is an implementation notion - what is needed is an ability to infer the answer to the question "does description A subsume description B?", and classifying descriptions in a taxonomy from which the answer to this question can be read off is one way to implement it.
- realization - inferring which individuals are described by which descriptions just on the basis of descriptions known to apply to them (e.g., that the conjunctive description "an A which is a B" applies to an individual that is known to be an A and known to be a B - see the Technical Discussion on the Realizer).
- various inferences over Role Value Maps, including the fact that if "A subset B" and "B subset A" then "A = B".
- inference of Value Restrictions through Role Value Maps (if two Roles are known necessarily to corefer, then the restrictions on their fillers can be inferred to be the same); this is the basis for Fikes' "Active Role Value Maps".
- co-specification inferences - it should be possible for the system to be able to tell when it is no longer possible for two descriptions to apply to the same individual, based on disjointness and exhaustiveness assertions; it should, by the same token, be possible to tell when two descriptions must specify the same individual.
- truth/reason maintenance - it should be possible for a knowledge representation system to maintain the dependencies between assertions, where those dependencies arise either from inferences made by the representation system itself or by an outside agent. The ability to detect inconsistencies among assertions, in general, was thought to be beyond the scope of a representation language.

- o unification of descriptions
- o accessibility, as among contexts representing possible worlds

It was also nearly uniformly agreed that two types of practical mechanisms are desirable: 1) tools for interfacing the system to a data base or "the real world", and 2) an "escape mechanism", so that a user of the representation system can specify certain things in the language in which the representation system is implemented. This latter was stated to be particularly important, in that implemented systems are inevitably incomplete.

At least two kinds of things were felt not appropriate ground for a representation system to cover:

1. perceptual activities, like recognition, wherein an "appropriate" description is determined for some individual. The representation system should be able to record the results of perceptual recognition and contribute information to the mechanism doing the recognition, but it is not the job of the representation itself to do such processing.

2. color graphics, etc.

Finally, there were a few more things not particularly inferential in nature that were expected to be present in an off-the-shelf knowledge representation system. These included basic concepts and axioms, like concepts for numbers; the most general description, from which all other descriptions are formed (e.g., the KL-ONE Concept "THING", with its one Role "subpart", from which all other Roles descend); concepts for LISP functions; and a self-description that is understood by the system itself (wherein an activity of the system can be made to happen by describing that activity in the language itself and causing it to "take").

2.3 Usage of contexts for representation of time and belief

Chaired by David Israel (BBN)

The notion of a CONTEXT (or SITUATION or POSSIBLE WORLD) seems to be necessary in specifying the semantics of a large class of important - so-called "intensional" - constructions. Such notions were first used to explain the meaning of the modal constructions - those in which the concepts of necessity and possibility played central roles. Intuitively, to say that a proposition is a necessary truth is to say that it's true in all possible worlds; to say that it's false but might be true is to say that in the actual world it's false, but that there are possible worlds (possible situations) in which it's true. The same idea has been used in explicating the meaning of counterfactuals - sentences of the form: If it were (had been) the case that P, then it would be (have been) the case that Q. Others (most importantly in AI, Bob Moore) have attempted to use the same kind of notion in explicating the logic and meaning of propositional-attitude constructions, such as: S believes (desires, hopes, wants, fears,etc.) that P. And a number of researchers have used similar ideas in handling temporal constructions (It will be the case that P; It used to be the case that P; but is now no longer. Etc.)

No claim is being made that exactly the same ideas are at work in all this work; but there are strong family resemblances among them. And central to them all is the introduction of a domain (a set) of contexts whose intrinsic structures are left largely or completely unspecified. Since theories of possible world-type semantics leave us so in the dark about the properties of these, they had best tell us something about the relations holding among them. And that's what most such accounts do: crucially, they specify a set of dyadic relations among worlds, situations, whatever, which are supposed to govern "movement" between these. In the case of the modalities, these relations are termed relations of relative accessibility or reachability; in the case of temporal constructions one might imagine, e.g., a relation of precedence among SITUATIONS or TIME-SLICES (or INSTANTS).

Let's assume that we are all convinced that we want to handle the kinds of constructions exampled above in KL-ONE; what then? Both the design and the implementation of such a facility raises significant problems. One that, in some sense, takes priority, can be put thus: do we go extensional or do we stay intensional? All of the constructions alluded to above are intensional; a natural way to treat them (and the way that in modal and tense logics and the logics of knowledge and belief they are, in fact treated) is to introduce intensional operators and either axioms or/and rules of inference governing these. That is, to introduce a new logic (or logical theory.) One can, however, avoid this move and handle these constructions in a standard, extensional (first-order) quantificational way by slightly altering the language one starts with in certain simple ways (and correlatively adding new structures to the original domain). So for instance, this is how both Moore and (though not so completely) McCarthy and Hayes go about things. In KL-ONE, this would require having a generic concept for SITUATIONS (INSTANTS, POSSIBLE WORLDS, what have you) and in conformity with the spirit of the thing, specifying a theory of worlds, etc. for which some KL-ONEish version of a sound (and complete?) proof-procedure for quantification theory was specified. NOTE, this way, at least in its pure form, requires no new kinds of structure (especially not if you think of KL-ONE as some oddly shaped notational variant of a standard quantificational language.) It does require new and special axioms dealing with possible worlds and relations among them and their "inhabitants". But this is up to the applications designer, in just the same way that it would be if the theory at hand were an arithmetic or biological (as opposed to a metaphysical) one.

The other way - and of course there are many variants here - surely will require new kinds of KL-ONE structures and new problems about their interactions with each other and with other KL-ONE structures (e.g. and most famously, perhaps: individual concepts and/or nexuses).

A major issue of discussion at the session was on the significance - within KL-ONE - of the difference between these two kinds of treatment. Was it a matter merely of semantic net design? (There were also heated exchanges on the different kinds of context-dependence with which, e.g., a natural language understanding system would have to deal.) There was, as well, explicit recognition of the ways in which these issues connected with issues about the treatment of propositions in KL-ONE. But, alas, there was no clear consensus.

3. PAPERS FOR PRESENTED TALKS

Several formal talks were presented at the Workshop. The papers in this chapter correspond to those talks. The series of talks that occurred on Monday night (entitled "Talks from various sites") are not reported here as they were intended to be informal descriptions of various projects. However, all other talks are represented by papers in this chapter. The first paper in this chapter was presented during the technical discussions. The others were presented at the main conference (each paper is noted as to when it was presented).

Please note that the section numbering scheme within each of these papers is disjoint from that of this proceedings. Whatever scheme was selected by each author was left intact, even though it may appear to overlap the section numbers assigned to other portions of the proceedings.

Realization

William Mark

USC/Information Sciences Institute

(Presented during the technical discussions)

1. Introduction

All knowledge based systems have as a key task the problem of "recognition", that is, relating a new piece of knowledge to the existing knowledge base. In KL-ONE, there is a strict separation between the intensional characteristics of representations in the knowledge base and their extensions in the real world. Intensional information is kept in the form of a taxonomy of concept definitions. Extensional information consists of the association of descriptions with real world objects and environments. Therefore, the recognition process can be viewed as consisting of two parts: use of intensional information to fit the incoming description into the definitional hierarchy, classification[1]; and use of extensional information to associate the incoming description with the appropriate real world entities, realization.

1.1. Definition of the Problem

The realization problem is defined by the following "principles":

- a description describes all real world entities described by its subconcepts;
- the only way to tell whether a description describes real world entities other than those described by its subconcepts is to use extensional knowledge;

[1]See Tom Lipkis' paper "A KL-ONE Classifier" on page 128.

o unless defined to be true at creation time, the coreferentiality of two descriptions (i.e., the fact that they describe the same real world entity) must be determined by extensional realization.

The first principle says that the realizer can find all of the real world entities that any description is defined to describe simply by looking down its subconcept chain. As each real world entity is created (by the action of processes external to the KL-ONE formalism), it will be given an initial description. By following the relationships defined in the knowledge base (and maintained by the classifier), the realizer can determine which of these initial descriptions--and thus which real world entities--are encompassed by any description of interest.

But the second principle warns that this reasoning is of limited utility. It will often be the case that descriptions of interest describe real world entities of interest not "by definition" (according to intensional knowledge), but "by circumstance" (depending on extensional knowledge). For example, knowing that "message that has been seen" describes the same real world entity described by "message 17" requires an appeal to extensional knowledge.

The final principle is really a simple corollary of the other two--but it emphasizes an important aspect of the problem. Many descriptions contain constraints involving coreferentiality as part of their definitions. For example, an "operation composition" of two operations requires in its definition that the input of one operation be coreferential with the output of the other. This means that in order to determine if some description represents an operation composition, the classifier must ascertain (among other things) that its outer operation has as input the output of its inner operation. But unless the classifier knows by definition that this is in fact the case, in short, unless the description was created as a subconcept of the operation composition description in the first place, the classifier can never make this determination. Realization using extensional knowledge is necessary.

Realization must therefore be used not only when a reasoner must find real objects in order to perform some task, but also whenever the reasoner is trying to determine the relationship between two descriptions when coreferentiality constraints are involved (i.e., whenever a description contains an RSR). Therefore, assuming that we wish to go beyond definitional reasoning--or assuming that we are incapable of representing everything we want in the definitional network--realization using extensional knowledge is a fundamental part of the reasoning process, complementing classification at all stages.

1.2. The Need for Simplification

The real world is a complex place. Reasoners must deal with partial knowledge, belief, context-dependent and context-independent information, information about individual objects, information about groups of objects, etc. Descriptions of all of these types certainly have bearing on the realization problem (any description _could_ describe any real world entity--or influence decisions about the description of any entity--and enough computation could conceivably discover that fact). As a practical matter, realization must be restricted due to the incompleteness of the current KL-ONE formalism and to the need for conceptual and computational tractability.

Any realization algorithm in the current KL-ONE environment must be an approximation. KL-ONE has no methodology for expressing and asserting propositions. Its context mechanism provides an implementation framework, but no restrictions on the interpretation and use of contextual information. The nexus construct has known limitations. Realization must work around these shortcomings until the formalism is complete.

Even in a complete formalism, realization will inevitably be approximate; the conceptual/computational problems of a complete scheme are too formidable. Every time a new description enters the system, the realizer must decide which real world entities must be examined, which existing descriptions have to be taken into account (the system could have widely dispersed information about contexts, partial knowledge, negative information), which existing realizations have to be changed, etc. This would require a seemingly boundless use of resources. Simplifying assumptions must be made if realization is to be feasible.

The next section describes the realizer implemented in the Consul system. I will outline the approximations and restrictions that allow it to work in current KL-ONE (and to work at all) and give examples of its use. Following sections examine the advisability of making realization part of the "standard KL-ONE system" and discuss the possibility of extending the current scheme in the direction of total realization.

2. The Consul Realizer

Given that the general realization task is of arbitrary complexity, the _practical_ problem of realization becomes one of defining a process that works within the current limitations of the KL-ONE representation, is limited enough to be computationally feasible, performs a useful (if not complete) range of reasoning tasks, and is clean enough to allow modification and extension.

The Consul realizer is an attempt to meet these goals. It is designed to support Consul's major inferential processes: mapping, explanation, and acquisition. It uses the KL-ONE representation to the fullest extent possible; where it must go beyond KL-ONE, the knowledge it uses is carefully isolated, packaged up, and attached to the appropriate parts of the KL-ONE representation. To provide computational feasibility, it is invoked only under certain well defined conditions. The result is that under a restricted set of circumstances, it can determine whether a restricted set of descriptions describe a restricted set of real world entities, given a rather restricted view of the real world. In particular:

- o <u>Restrictions on Invocation</u>: The realizer is called only when the classifier adds a new description to the network, and even then only when the incoming description has certain characteristics (see below).

- o <u>Restrictions on Descriptions Investigated</u>: Only a small set of descriptions (the "realization set") is considered during each invocation:

  - . the incoming concept;
  - . superconcepts of the incoming concept;
  - . concepts that would be subconcepts or superconcepts of the incoming concept except for the presence of non-preempted RSR constraints;
  - . all other descriptions of nexuses for which a new description is found during the invocation.

- o <u>Restrictions on Extensional Knowledge Used</u>: All of the extensional knowledge that can be used to perform the realization must either be part of the realizer code (i.e., knowledge about how to process certain network structures with no reference to the "meaning" of the concepts in those structures) or be represented as procedures attached to concepts that are expected to be paraindividuated (these procedures define the "meaning" of that particular concept).

- o <u>Restrictions on World View</u>:

  - . The only way to say something about the real world is to assert the description of a nexus in some context. No other descriptions (e.g., general propositions, descriptions of environments and configurations of nexuses) can be asserted (see Section 1.1, page 8).
  - . A nexus is a simple describable object (there are no variable nexuses).

- A context is an arbitrary grouping of nexuses. The realizer knows nothing about the semantics of contexts (e.g., how one context relates to another).

- The world of nexuses is considered to be closed: it is assumed that every real world object that should be represented by a nexus is represented by a nexus. This is accomplished by processes outside the realizer.

- For a given invocation of the realizer, only those nexuses which are described by descriptions in the realization set are considered.

- The realizer makes no decision about the merging of nexuses.

2.1. The Realization Algorithm

The action of the realizer is as follows:

(1) after the classifier classifies an incoming description, it calls the realizer with a list of pairs, each consisting of the incoming description and (a) one of the "almost subconcepts" of the incoming description that do not have some RSR's that are on the incoming description, (b) one of the superconcepts of the incoming description that do not have some RSR's that are on the incoming description, or (c) one of the "almost superconcepts" of the incoming description that have RSR's that are not on the incoming description; the members of each pair are checked by the realizer for a possible subdescription/superdescription relationship (the incoming description may be either the subdescription or the superdescription of the pair);

(2) the realizer first checks to see whether the possible subdescription happens to describe the nexuses described by the possible superdescription:

- o if the possible subdescription contains non-RSR information not contained on the possible superdescription (e.g., a more restrictive VR for one of its roles), the realizer cannot determine whether it describes the nexuses described by the possible superdescription; no further checking is done, and the pair is discarded;

- o otherwise, the realizer checks to see whether the RSR constraints on the possible subdescription are in fact satisfied by the nexuses attached to the possible superdescription; if they are, there is indeed a subdescription/superdescription relationship, and the nexuses of the superdescription are attached to the subdescription;

(3) the realizer then checks to see whether the possible superdescription describes the nexuses described by the possible subdescription:

- o if the possible superdescription is an actual superconcept of the possible subdescription, it is automatically known to describe the nexuses described by the subdescription, and nothing further needs to be done;

- o otherwise, the realizer checks to see whether the RSR constraints on the possible superdescription are in fact satisfied by the nexuses attached to the possible subdescription; if they are, there is indeed a subdescription/superdescription relationship, and the nexuses of the subdescription are attached to the superdescription;

(4) every time a new description wire is created, the realizer finds all of the existing descriptions of the nexus involved; if there are more than one, it creates the common subconcept of all existing descriptions; this new description must also describe the nexus, so it is appropriately attached; the new description is then classified in the network.

"Satisfaction of RSR constraints" with respect to a particular nexus is determined by checking to see whether its attached descriptions accord with the structural relationship expressed by the RSR and (in some cases) with procedural specifications for the paraindividuals involved. Constraints involving only "further description" and role value maps can be checked structurally--the meaning of particular paraindividuals is not in question. In all other cases, the meaning of the paraindividual must be considered, and is represented via an attached procedure (i.e., not in KL-ONE). Also included in each constraint is the number restriction on each role coreferenced by the RSR: it tells the realizer how many nexuses the value description of that role is allowed to describe.

2.2 An Example of Realization

This may be easier to unravel in the context of an example (see Figure 1). Suppose there are three nexuses described as messages; two of the

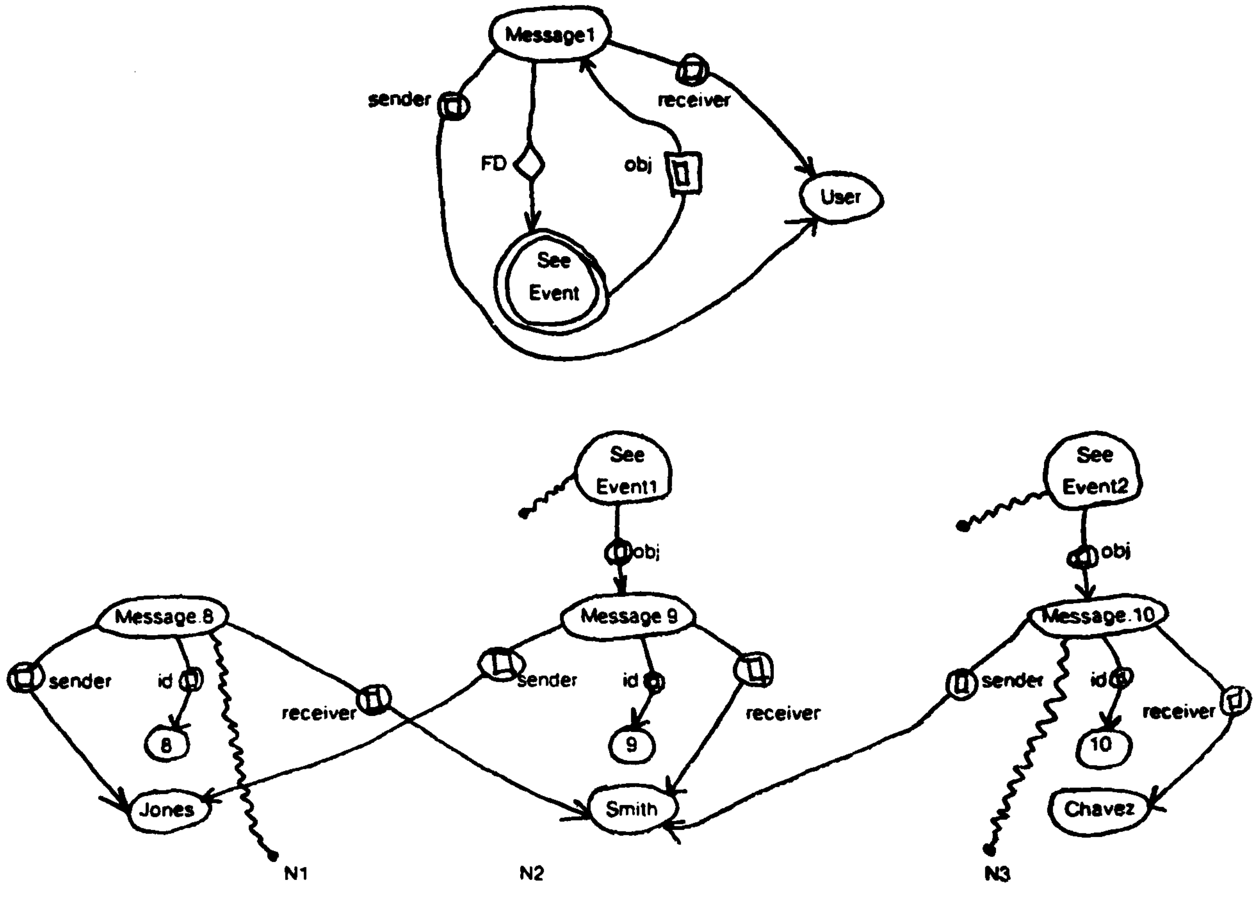
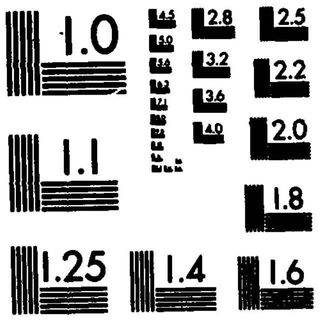


FIG. 1. A NEW DESCRIPTION IS INTRODUCED INTO THE KNOWLEDGE BASE

descriptions are also value restrictions of See Event's. Now we introduce a new description, Message1: "messages that have been seen". In the course of classification, this incoming description will be compared with Message.8, Message.9, and Message.10. Since Message1 would be a superconcept of the other message descriptions except for the SD See Event, Message.8, Message.9, and Message.10 are "almost subconcepts" of Message1. Following step (1) of the algorithm, the "possible subdescription"/"possible superdesription" pairs (Message.8 Message1), (Message.9 Message1), and (Message.10 Message1) are passed to the realizer.

AD-A123 772 PROCEEDINGS OF THE 1981 KL-ONE WORKSHOP(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA J G SCHMOLZE ET AL. JUN 82 BBN-4842 N00014-77-C-0378 2/3

UNCLASSIFIED F/G 5/6 NL



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

For each pair, the realizer first attempts to determine whether the possible subdescription describes the nexuses described by the possible superdescription (step (2)). In each case, this is prevented by the fact that there is additional non-RSR information on the subdescription (the VR's of the sender and receiver roles, the additional id role).

The realizer must next see whether the possible superdescriptions describe the nexuses described by the possible subdescriptions (step (3)). This means determining whether the constraint expressed by the "further description" See Event is true of the nexuses described by Message.8, Message.9, and Message.10 (respectively). Further description implies that a subconcept of the paraindividuated concept has a role with a value as indicated by the coref role of the paraindividual. That is, the further description constraint expresses a template that can be checked in the network structure. In this case, the template is satisfied if the Message in question fills the object role of some kind of See Event description.

This notion of "filling the object role" must be examined carefully. In KL-ONE, the fact that a Message description is the VR of the object role of a See Event description says nothing about whether a particular entity (nexus) described as that Message is in fact in an "object" relationship with the particular entity described as a See Event. This would only be true if we could somehow show that the description Message uniquely describes that particular entity, i.e., that that description could not possibly describe any other nexus, so it must mean this one. In short, for the realizer to check the further description constraint, it must see whether the Message description is individual in this way under these circumstances[2].

In this case, Message.8 is determined to be an individual description. It is then checked to see whether it is the value of the object role of some kind of See Event that describes a nexus; it is not. The further description constraint therefore does not hold, and Message1 does not describe nexus N1. However, similar reasoning for Message.9 and Message.10 leads to the conclusion that Message1 does indeed describe nexuses N2 and N3. The realizer therefore attaches Message1 to N2 and N3.

I will postpone discussion of step (4) of the algorithm and go on to Figure 2, showing the introduction into the knowledge base of another new description, Message2: "messages from Jones to Smith when both were logged on". In the previous case, the incoming description was the possible

---

[2]This argument, as well as the system's mechanism for determining individuality, is described in much greater detail in Mark's summary of the technical discussion on "Individuality in KL-ONE", Section 1.3, page 23).

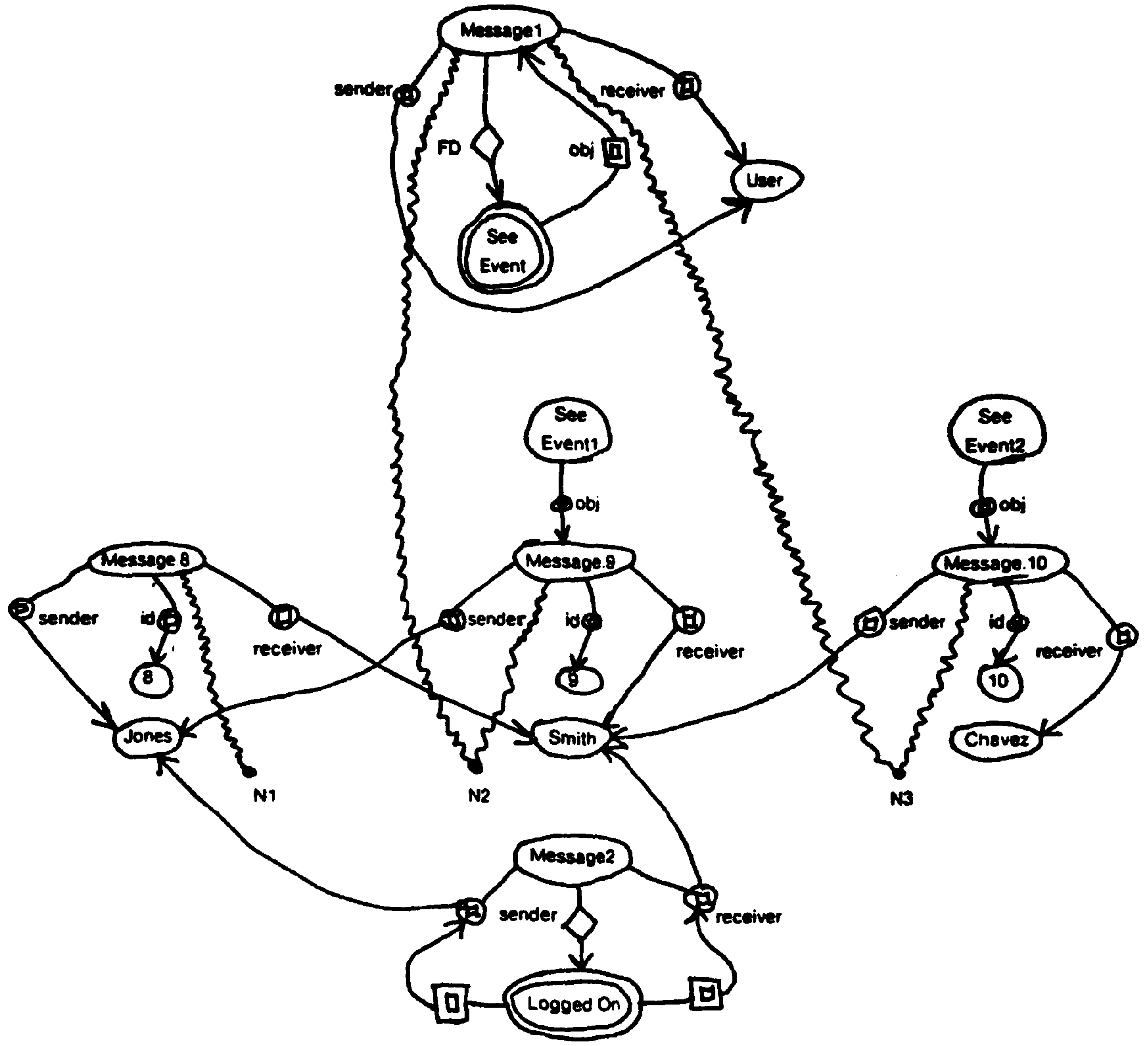


FIG. 2. ANOTHER NEW DESCRIPTION IS INTRODUCED INTO THE KNOWLEDGE BASE

superdescription of the existing descriptions Message.8, Message.9, and Message.10, thus exercising step (3) of the realization algorithm. In this case, Message2 is a possible subdescription of Message.8 and Message.9, so that step (2) is the one that is applicable. Realization processing takes place much as before except that the SD Logged On is not a "further description", and requires different processing.

The realizer must essentially determine whether the constraint expressed by Logged On is true of the nexuses described by Message.8 or Message.9. The desired constraint is that the message have the characteristic that both its sender and receiver were logged on when it was sent. The question is how this can be represented and how the realizer can determine whether or not it holds in this case. I will make the assumption that the constraint is not easily represented in terms of the current KL-ONE formalism, and must therefore somehow be represented extrinsically, i.e., outside the world of network and nexuses. The problem then becomes to appropriately package extrinsically represented constraints so that they can be accessed and checked by the realizer in a consistent manner.

The mechanism chosen in the current realizer is to express all constraints except "further description" SD's and role value maps as procedures attached to the relevant generic concepts. When these concepts are paraindividuated, the constraint procedure is inherited. Then when the paraindividual becomes involved in a realization, the realizer accesses the procedure, evaluates it on the relevant arguments (as determined by the coref roles of the SD), and returns true or false. Thus, in this case it could determine that, say, the Logged On constraint was true of Message.9 but not Message.8.

Note that there is nothing in Figure 2, in other words, nothing in the KL-ONE representation of that situation, that indicates why Message2 describes the nexus described by Message.9 but not Message.8. For further description constraints, it must be the case that the relevant nexuses have been explicitly created and appropriately described. For procedural constraints, all of this knowledge can be implicit (i.e., "somehow" known to the procedure, and to nothing else).

This is certainly undesirable: we want to move in the direction of the explicit structure-based reasoning possible now only for constraints that express nothing more than coreferentiality relationships, i.e., only for further descriptions and role value maps. Extension of the scheme to more complex predicates awaits the new proposition representation (see the Section "Extensions", page 20, in the summary of the technical discussion on Realization).

There is one more set of nexuses that can be found by the realizer, as defined in step (4) of the algorithm description. It is illustrated in Figure 3. The upper part of the figure shows a nexus N4 described as both a red haired girl and a cyclist. Now the new description Cyclist'--a red haired cyclist--is introduced. The realizer should be able to discover that it too describes nexus N4. The problem is that there is no single concept in the

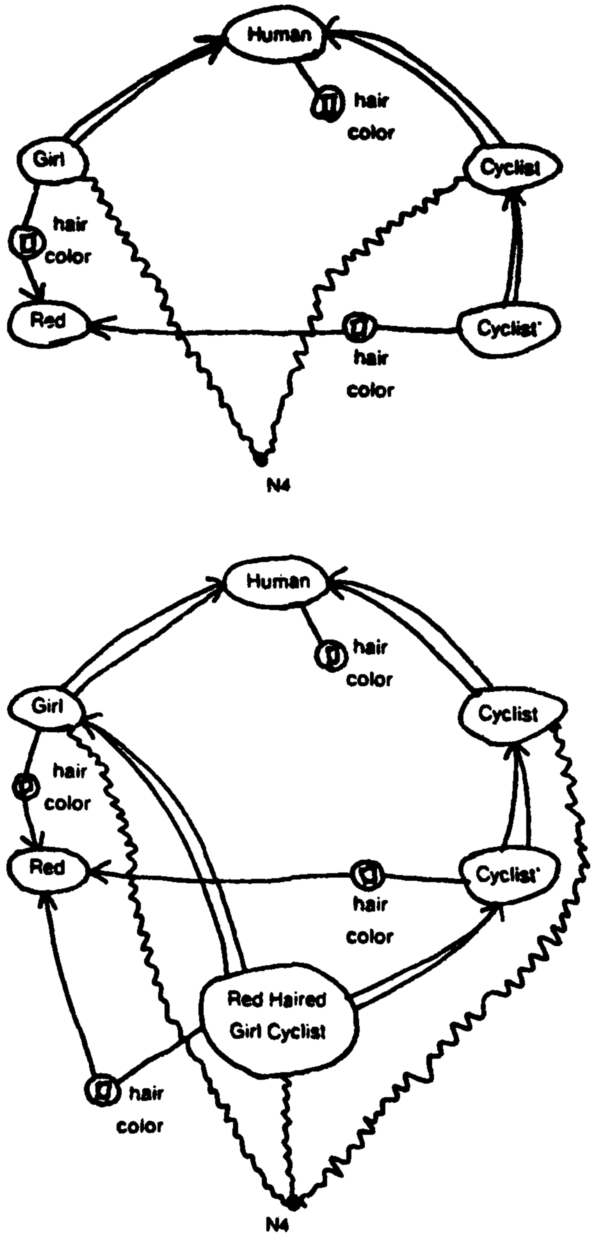


FIG. 3. FINDING NEW REALIZATIONS BY COMMON SUBCONCEPT CREATION

network that contains the information necessary to make this determination.

The necessary information is in fact the conjunction of all concepts known to describe N4. If the realizer is to discover this sort of realization, it must somehow be aware of the different pieces of information in the network that can be taken conjunctively.

This capability is provided in the current realizer by prior construction of the common subconcepts: whenever the creation of a new description wire produces a multiply described nexus, the common subconcept of all the descriptions of that nexus is created and attached to the nexus. This method was chosen to simplify the action of the realizer--it allows this case to be handled naturally in the framework of realization as an adjunct of classification. If common subconcept creation were left implicit or made explicit only on some sort of "need" basis, that is, if common subconcepts were not always explicitly represented in the network, classification could not be relied upon to find all realization opportunities.

In this case, when it was discovered that both Cyclist and Girl describe N4, the description Red Haired Girl Cyclist was produced. When (some time later) the red haired cyclist Cyclist' is introduced, Red Haired Girl Cyclist will be a subconcept of it, and Cyclist' will be seen to describe N4 by subconcept link examination. This situation, the actual one produced by the realizer, is shown in the lower half of Figure 3. This is another example of structural reasoning by the realizer: there is no appeal to extrinsic knowledge.

In the examples shown in Figures 1 and 2, each multiply described nexus has one of these common subconcept descriptions like Red Haired Girl Cyclist (e.g., N1 would have the additional description "message from Jones to Smith with id 8 that has been seen). However, since no new realizations are revealed by the resulting subconcept relationships, these common subconcept descriptions are not shown (the diagrams are complicated enough as it is).

Imagining the presence of the appropriate common subconcepts and a description wire from N2 to Message2, Figure 2 shows the complete result of realization on the examples.

Highlights from KloneTalk: Display-Based Editing and Browsing, Decompositions, Qua Concepts, and Active Role Value Maps

Richard Fikes

Cognitive and Instructional Sciences Group
Xerox Palo Alto Research Center
Palo Alto, California

(Presented during the main conference)[1]

This presentation describes some of the distinctive features of an implemention of KL-ONE in Smalltalk [6] called KloneTalk. The motivation for this implementation came from some work done by Austin Henderson and myself involving the development of systems that understand how procedural tasks are actually done in an office rather than how they are described in policy and procedure manuals [1 and 4]. We were interested in developing a terminology for describing office tasks, functions, and procedures and we needed a formal language in which to express that terminology. We decided to use KL-ONE as the formal language and to develop a set of system facilities for creating, editing, extending, and browsing KL-ONE networks that would support the terminology development. We wanted a display-based interface for our system and the capability to extend KL-ONE itself when necessary to provide the representational facilities necessary for our project. These requirements led us to build our own implementation of a subset of the KL-ONE language.

1. THE KLONETALK USER INTERFACE

KloneTalk presents the user with a display-based interface containing facilities for creating, editing, extending, and browsing KL-ONE networks. In addition, there is a file package that allows the user to read and write portions of a network onto text files in a "pretty print" format that is human readable and editable "off line" using a text editor. This section is an overview of that user interface. (For a more detailed description, see [2]).

1.1. The Network Index Window

[1]A videotape was shown which demonstrated the user interface to KloneTalk.

For each KL-ONE network (typically only 1) in the system, the user can display a window containing an index into that network. The index consists of four lists of names, alphabetically ordered. The lists are of generic concepts, individual concepts, nexuses, and contexts. Figure 1[2] shows an index for a small network.

(Note: We have focused our attention in the implementation almost entirely on generic and individual concepts, to the exclusion of nexuses and contexts. Hence, even though nexuses, contexts, and description wires can be represented in KloneTalk, we have given no attention to providing user interface facilities for their convenient usage.)

The user can select any item on these lists and then obtain a menu of operations for the selected item. For any of the four types of items, those operations include Rename, Remove, Spawn View, and Spawn Full View. For generic concepts, the operations also include Specialize, Typify, and Individuate.

The network index window is intended to be used primarily to initiate a series of interactions with a network. Most succeeding interactions will be done from item viewing windows in a browsing mode, as described below.

1.2. Item Viewing Windows

When the user selects the "Spawn View" operation, a new window is opened on the screen at a location and with a size specified by the user with the mouse. A description of the item to be viewed (e.g., a generic concept) is then created using a simple parenthesis language and displayed in the window as a text string (Figure 2 shows an example of a view of a generic concept). The user can then edit that description using the standard Smalltalk text editor. The menu of edit commands includes "Compile", and when that command is selected, the item described by the edited text is stored in the network. If that item has the same name as an existing item in the network (as it would in the typical case where the edits do not change the item's name), then the old item is replaced by the new item in such a manner that all other items in the network that pointed to the old item now point to the new item (more about how that is done later).

The same menu operations that are available in the network index window are also available in an item viewing window. For example, when a generic concept is being viewed, the user can specialize or individuate it. If the

[2]All figures are included at the end of this paper.

name of an item is selected in a viewing window, then the menu of operations applies to the selected item rather than to the item being viewed. So, the user can browse the network by selecting the name of any item mentioned in the description (say, as the value restriction of a role), and then use the SpawnView menu command to create a view of the item mentioned.

It is often very useful when viewing concepts to be able to see descriptions of the inherited information in addition to the local information. The "Spawn Full View" command provides that capability. In a full view of a concept, the local information is displayed in bold and the inherited information in non-bold (Figure 5 shows an example of a full view of an individual concept). The user can edit a full view, changing the edited sections to bold, and compile it. During the compilation, the parser ignores all non-bold characters.

The full view provides a complete description of the concept. Such a description is particularly useful as an editing template when new concepts are being created, since it indicates the rolesets that are available to modify or differentiate. A standard way of extending a network is to use a Specialize or Individuate command to create a new concept, spawn a full view of the new concept, and then describe the concept by editing the full view.

The interface contains many detailed convenience features not described here. One worth mentioning is an "Information" menu command that applies to an item being viewed or whose name has been selected in a view. That command results in display of an additional menu of commands that can provide the user with various data about the item that is not included in the text description. Such information primarily involves identifying other items that point to this item. For example, for a generic concept there are commands to list the roles of which the concept is a value restriction, the concept's specializations, and the concept's individuals. This information menu (for any item) also contains a hook into Smalltalk's facilities for viewing Smalltalk data structures. That "Inspect" command allows the user to view the data structure that represents the item for debugging or system modification purposes.

1.3. Automatic Definition of Mentioned Concepts

The compiler will define any item mentioned in a description that does not yet exist in the network. So, for example, if a role's value restriction is not already in the network, then it will be created as a new generic concept specializing Entity (the most general concept in our network). The compiler uses whatever information it has from the context when creating new items. So, for example, a role's value will be created as an individual concept individuating the role's value restrictions.

This compiler capability of defining an item whenever its name is mentioned is very convenient for the user when adding concepts to a network, and allows items to be mentioned in descriptions on a file before the mentioned item is described on the file. Some protection is needed for the user, however, from the system creating a new item when the name of an existing item is mistyped.

1.4. Internal Methods for Changing the Network

Care must be taken in designing the compilation methods for changing an existing item in a network. For example, when the user asks to view a concept, edits the description of the concept, and then issues the Compile menu command; if he has not changed the name of the concept, we assume that he intends the existing concept in the network to be altered so that it matches the edited description. That alteration must be done in a manner so that any other items in the network that reference the old concept or any of its parts continue to reference the appropriate places in the new concept.

The methods we have implemented reuse the same data structures for concepts, roles, structural descriptions, nexuses, and contexts so that pointers to those structures are still valid. (That is, we assume that if one of those items has the same name in an edited description as it had before the edit, then the user intends all references to the old item to now be references to the new item.) The difficult problem then becomes one of dealing with references to items deleted by an edit. For example, during the edit of a concept, a role may be deleted that is differentiated by a role of a specialization of the edited concept.

We have organized our implementation so that whenever an item is being removed, a "remove" function for that type of item is called (hence, there is in effect a RemoveRole function, a RemoveGenericConcept function, etc.). Our current versions of those functions take a zeroth order approach to the problems by simply insuring that all pointers to the item being removed are also removed from the network. For example, if a generic concept is a value restriction of some role and is removed from the network, then it is also removed from the role's list of value restrictions. These functions therefore maintain syntactic consistency in the network, but provide no help in assuring reasonable semantic consequences of the removals.

My intuition is that desirable versions of these removal functions would inform the user when references to the item being removed are found in the network, and, whenever possible, present reasonable options of what to do with them in addition to deletion. For example, when a generic concept is being removed that has specializations, one might present the option of altering the specializations so that they become specializations of the concepts that the removed concept specialized.

2. ADDITIONS TO KL-ONE

In the course of using KloneTalk, we have experimented with several additions to KL-ONE. This section discusses two of those that have proven particularly useful: decompositions and qua concepts.

2.1. Decompositions

In our use of KloneTalk, we wanted to be able to include in the definition of concepts statements of the form "all X's are either Y's or Z's." and "an X cannot be a Y". To provide that capability, we add facilities that allowed the definition of a generic concept to include a collection of decompositions, each of which specifies a way in which the concept's specializations decompose it.

Each decomposition consists of a set of constituents, a disjointness flag, and a completeness flag. Each constituent is a generic concept that is a specialization of the concept being defined. For example, a "Person" concept might have an "age" decomposition consisting of "Child" and "Adult" that is complete and disjoint (i.e., is a partition). Given this decomposition, the system could conclude that if N is a Person, then N must be a Child or an Adult, and if N is a Child then N is not an Adult. Also, the "Person" concept could have an "AdultsBySex" decomposition consisting of "Man" and "Woman" that was disjoint but not complete (i.e., does not include children). Given that decomposition the system could not conclude that a Person must be either a Man or a Woman, but could conclude that a Man is not a Woman.

2.2. Qua Concepts

We have included in KloneTalk a version of "Qua" concepts (first suggested in [5]). In our implementation, each role at each concept has an implicit qua concept associated with it. Qua concepts allow the description of characteristics of the value of a role that the value has only because it is the value of that role. For example, the "wife" role of a "Marriage" may have "Woman" as its value restriction. But a woman who is a wife could be considered to have roles such as "husband", "mother-in-law", and "maidenName" that she acquires only because she is the wife of a marriage. Hence, those roles would be defined as being part of the definition of the qua concept for "wife" at "Marriage". The qua concept for a role is by definition a subconcept of the role's value restriction. Hence, in our example, the qua concept for "wife" at "Marriage" would be a subconcept of "Woman".

Qua concepts have the following implicational import in the assertional portion of KloneTalk. Let Q be the qua concept for role R at concept GC. Then, describing nexus N1 as a Q in a context implies the existence of a nexus N2 in the same context describable as a GC whose R value also describes N1 (i.e., being Q'ish implies being R'ish for some GC).

In our implementation, a Qua concept for a role is simply a generic concept with an additional "qua link" to the role. In addition, it is required to be a subconcept of the value restriction of the role and, if the role is inherited from some concept, of the qua concept of the role at that concept.

3. ACTIVE ROLE VALUE MAPS

Several members of the KL-ONE community have found "Role Value M -" (RVMs) to be a particularly useful subclass of roleset relations. We found in our work with KloneTalk that their usefulness extends to aiding building and editing of concept networks as follows. When new qua individual concepts are added to a network, information implicitly represented in RVMS of concepts already existing in the network can be explicitly added to the description of the new concepts. The concept description additions involve differentiating roles at the end of role chains and adding values to those differentiations, or adding new value and number restrictions to roles at the end of the role chains. Such "active role value maps" relieve the user of the need to supply that information himself.

This section describes the syntax and semantics of RVMs as they are implemented in KloneTalk, and then provides an example of how they are used to automatically add information to qua and individual concepts. (For a more detailed description, see [3].)

3.1. RVM Syntax and Semantics

A RVM in KloneTalk has the form:

```
<RVM> := <role chain> <connective> <role chain>
<role chain> := {Each} <role id> |
                <role chain> {Each} <role id>
<connective> := < | = | >
```

A role chain specifies a sequence of role identifiers (<concept name, role name> pairs in KloneTalk) whose first element is a role of the concept that the RVM is attached to. It defines a tree of paths, since any role in the sequence may be differentiated.

Each path through a role chain tree leads to a set of roles (called the "end roles") that are differentiations with number restriction 1 of the last role in the path. A path also specifies a set consisting of the values (called "end values") of the end roles. The connective refers to the sets of end values and is interpreted as either "=" (set equals), "<" (subset of), or ">" (superset of).

When a role chain contains no occurrences of "Each", the RVM applies to the union of the sets of end values for the entire role chain tree. For example, a concept Parentage might have an RVM:

```
  (child from: Parentage) < (father from: Parentage)
                            (child from: Father)
```

meaning that the children of a parentage (i.e., of a particular coupling) are a subset of the children of the father. An occurrence of "Each" preceding a role id in a role chain means that the RVM applies successively to each subtree defined by each differentiation of the role indicated by the role id. For a given subtree, the constraint refers to the union of the set of end values in that subtree. For example, the concept Parentage might have an RVM:

```
 (father from: Parentage) = Each (child from: Parentage)
                                 (father from: Child)
```

meaning that the father of a parentage is the father of each of the parentage's children.

3.2. Examples of RVM Activation

Our methods for applying an RVM proceed by forming descriptions of the end value sets for the two role chains and then considering each pair of end value sets that are being constrained. For each role chain, the set being constrained is the set of possible values for the roles at the end of the chain. Changes can be made in such an end role if all the roles on the chain that led to it either had values or had qua concepts defined for them. Hence, an RVM can only be used to change the description of roles of individual concepts and qua concepts.

Rather than describe in this summary the details of the activation algorithms, I will illustrate their usage with a sequence of examples. Consider compilation in KloneTalk of the generic concept "Parentage" shown in Figure 2. If the Child, Father, and Mother qua concepts did not previously exist in the network, then the compiler would create them and give them the roles mentioned in Parentage, as shown in Figure 3.

Activation of the RVMs of Parentage would then cause the descriptions of those concepts to be augmented as shown in Figure 4. ParentageSD3 is used to imply that the value of the "mother" role of a Child must also be a value of the "mother" role of a Parentage, and since each Parentage as exactly one mother, that each Child has exactly one mother. ParentageSD4 implies the same results for the "father" role of a Child. ParentageSD1 is used to imply that a Father has at least one child and that at least one of those children must also be the "child" of a "Parentage". Similarly, ParentageSD2 is used to imply that a Mother has at least one child and that at least one of those children must also be the "child" of a "Parentage".

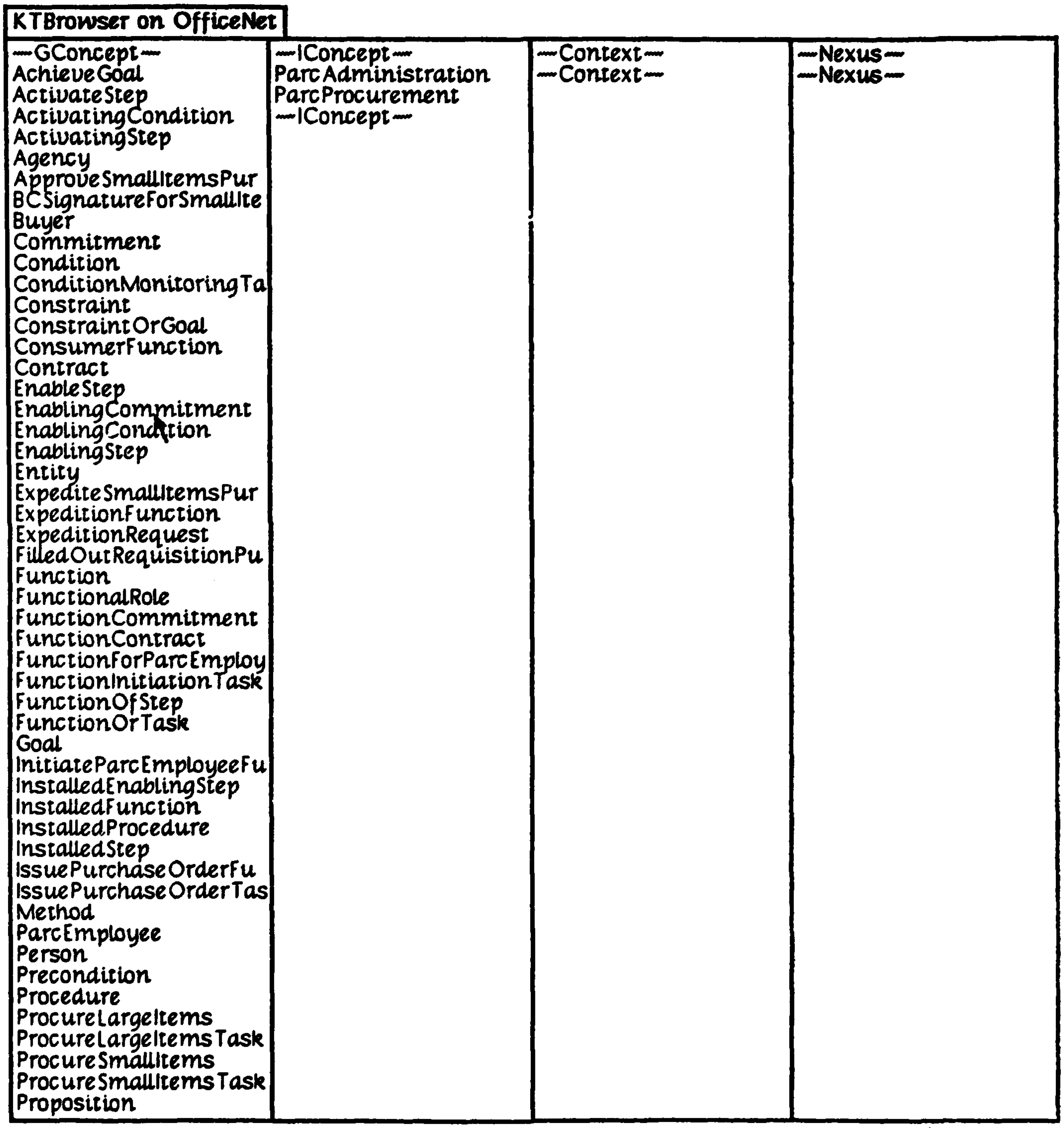
Consider the effects of compiling and applying the RVMs of an individuation of Parentage such as the one shown in Figure 5. If Child, Father, and Mother had the descriptions discussed above (and shown in Figure 4), and Joan, Jack, and Sue were not defined in the network before the compilation, then those individual concepts would be defined by the compiler and have the descriptions shown in Figure 6 after application of the RVMs.

Note that Joan could not be added as a child to the descriptions of Jack and Sue because the information is not available to determine whether Joan is an individuation of 'child' or of 'childSub1'. However, if before this compilation the user had edited the descriptions of Father and Mother, adding the information that the value restriction of the 'child' roles is 'Child' and deleting the 'childSub1' roles (so that every child is a Child), then application of the RVMs of ParentageOfJoan would cause the descriptions of Jack and Sue to appear as shown in Figure 7. In those descriptions, ParentageSD1 was used to determine that Joan is one of the children of Jack, and ParentageSD2 was used to determine that Joan is one of the children of Sue.

4. Acknowledgements

Several people have contributed ideas and effort to KloneTalk. Austin Henderson was a co-initiator of the project, a major contributor to the system's design, and a participant in the programming. Ron Brachman, Ira Goldstein, and Steve Weyer provided significant consulting support, and Ben Cohen participated in the programming.

FIG. 1. AN INDEX FOR A SMALL NETWORK



KTBrowser on OfficeNet

—GConcept—	—IConcept—	—Context—	—Nexus—
AchieveGoal	ParcAdministration	—Context—	—Nexus—
ActivateStep	ParcProcurement		
ActivatingCondition	—IConcept—		
ActivatingStep			
Agency			
ApproveSmallItemsPur			
BCSignatureForSmallIte			
Buyer			
Commitment			
Condition			
ConditionMonitoringTa			
Constraint			
ConstraintOrGoal			
ConsumerFunction			
Contract			
EnableStep			
EnablingCommitment			
EnablingCondition			
EnablingStep			
Entity			
ExpediteSmallItemsPur			
ExpeditionFunction			
ExpeditionRequest			
FilledOutRequisitionPu			
Function			
FunctionalRole			
FunctionCommitment			
FunctionContract			
FunctionForParcEmploy			
FunctionInitiationTask			
FunctionOfStep			
FunctionOrTask			
Goal			
InitiateParcEmployeeFu			
InstalledEnablingStep			
InstalledFunction			
InstalledProcedure			
InstalledStep			
IssuePurchaseOrderFu			
IssuePurchaseOrderTas			
Method			
ParcEmployee			
Person			
Precondition			
Procedure			
ProcureLargeItems			
ProcureLargeItemsTask			
ProcureSmallItems			
ProcureSmallItemsTask			
Proposition			

```
[Parentage (a: Relationship)
  (Text: 'The relationship resulting from conceiving a child')
  (RoleSets:
    (mother
      (ValueIsA: Woman)
      (Qua: Mother)
      (Number: 1))
    (father
      (ValueIsA: Man)
      (Qua: Father)
      (Number: 1))
    (child
      (ValueIsA: Person)
      (Qua: Child)
      (Number: (1 nil))))
  (RoleValueMaps:
    (ParentageSD1
      (Text: 'The children of a Parentage are some of the children of its father')
      (Map: (child from: Parentage) < (father from: Parentage) (child from: Father)))
    (ParentageSD2
      (Text: 'The children of a Parentage are some of the children of its mother')
      (Map: (child from: Parentage) < (mother from: Parentage) (child from: Mother)))
    (ParentageSD3
      (Text: 'The mother of a Parentage is the mother of each of its children')
      (Map: (mother from: Parentage) = Each (child from: Parentage) (mother from: Child)))
    (ParentageSD4
      (Text: 'The father of a Parentage is the father of each of its children')
      (Map: (father from: Parentage) = Each (child from: Parentage) (father from: Child]
```

FIG. 2. THE "PARENTAGE" GENERIC CONCEPT

```
[Child (a: Person)
   (Qua: (child from: Parentage))
   (RoleSets:
      (mother)
      (father]

[Father (a: Man)
   (Qua: (father from: Parentage))
   (RoleSets:
      (child]

[Mother (a: Woman)
   (Qua: (mother from: Parentage))
   (RoleSets:
      (child]
```

FIG. 3. CONCEPT DESCRIPTIONS RESULTING FROM COMPILATION OF PARENTAGE

```
[Child (a: Person)
  (Qua: (child from: Parentage))
  (RoleSets:
    (mother
      (ValueIsA: Mother)
      (Number: 1))
    (father
      (ValueIsA: Father)
      (Number: 1]
[Father (a: Man)
  (Qua: (father from: Parentage))
  (RoleSets:
    (child
      (Number: (1 nil)))
    (childSub1
      (Difs: (child from: Father))
      (Number: (1 nil))
      (ValueIsA: Child]
[Mother (a: Woman)
  (Qua: (mother from: Parentage))
  (RoleSets:
    (child
      (Number: (1 nil)))
    (childSub1
      (Difs: (child from: Mother))
      (Number: (1 nil))
      (ValueIsA: Child]
```

FIG. 4. DESCRIPTIONS AFTER ACTIVATION OF THE RVM'S OF PARENTAGE

```
[ParentageOfJoan (a: Relationship)
  (Individuates: Parentage)
  (Text: 'The relationship resulting from conceiving Joan')
  (RoleSets:
    (mother (from: Parentage)
      (Value: Sue)
      (ValueIsA: Mother)
      (Number: 1))
    (father (from: Parentage)
      (Value: Jack)
      (ValueIsA: Father)
      (Number: 1))
    (child (from: Parentage)
      (Value: Joan)
      (ValueIsA: Child)
      (Number: (1 nil))))
  (RoleValueMaps:
    (ParentageSD1
      (Text: 'The children of a Parentage are some of the children of its father')
      (Map: (child from: Parentage) < (father from: Parentage) (child from: Father)))
    (ParentageSD2
      (Text: 'The children of a Parentage are some of the children of its mother')
      (Map: (child from: Parentage) < (mother from: Parentage) (child from: Mother)))
    (ParentageSD3
      (Text: 'The mother of a Parentage is the mother of each of its children')
      (Map: (mother from: Parentage) = Each (child from: Parentage) (mother from: Child)))
    (ParentageSD4
      (Text: 'The father of a Parentage is the father of each of its children')
      (Map: (father from: Parentage) = Each (child from: Parentage) (father from: Child]
```

FIG. 5. AN INDIVIDUATION OF PARENTAGE

```
[Joan
  (Individuates: Child)
  (RoleSets:
    (mother (from: Child)
      (Value: Sue)
      (ValueIsA: Mother)
      (Number: 1))
    (father (from: Child)
      (Value: Jack)
      (ValueIsA: Father)
      (Number: 1]
[Jack
  (Individuates: Father)
  (RoleSets:
    (child (from: Father)
      (Number: (1 nil)))
    (childSub1
      (Difs: (child from: Father))
      (Number: (1 nil))
      (ValueIsA: Child]
[Sue
  (Individuates: Mother)
  (RoleSets:
    (child (from: Mother)
      (Number: (1 nil)))
    (childSub1
      (Difs: (child from: Mother))
      (Number: (1 nil))
      (ValueIsA: Child]
```

FIG. 6. DESCRIPTIONS OF INDIVIDUALS IMPLIED BY PARENTAGEOFJOAN

```
[Jack
   (Individuates: Father)
   (RoleSets:
      (child (from: Father)
         (Number: (1 nil)))
      (child1
         (Difs: (child from: Father))
         (Value: Joan]

[Sue
   (Individuates: Mother)
   (RoleSets:
      (child (from: Mother)
         (Number: (1 nil)))
      (child1
         (Difs: (child from: Mother))
         (Value: Joan]
```

FIG. 7. DESCRIPTIONS OF JACK AND SUE IMPLIED BY PARENTAGEOFJOAN, ASSUMING AN EDIT OF FATHER AND MOTHER

REFERENCES

[1] Fikes, R. (1981a)
"A Commitment-Based Framework for Describing Informal Cooperative Work". Proceedings of the Third Annual Conference of the Cognitive Science Society, Berkeley, CA, August.

[2] Fikes, R. (1981b)
"A Brief Sketch of the KloneTalk User Interface". Xerox PARC internal memo. October 1981.

[3] Fikes, R. (1981c)
"Active Role Value Maps in KloneTalk". Xerox PARC internal memo. October 1981.

[4] Fikes, R., and Henderson, A. (1980)
"On Supporting the Use of Procedures in Office Work". Proceedings of The First Annual National Conference on Artificial Intelligence, Stanford, CA.

[5] Freeman, M. W., and Leitner, H. H. (1981)
"Knet Extensions". Unpublished. Abstracted in Journal of Computational Linguistics, July-September, 1981.

[6] Ingalls, D. H. (1978)
"The Smalltalk-76 Programming System Design and Implementation". Conference Record of the Fifth Annual ACM Symposium on Principles of Programming Languages.

Translating KL-ONE from Interlisp to FranzLisp

Tim Finin

Franz KL-ONE translation project
University of Pennsylvania

(Presented during the main conference)

This section describes an effort to translate the Interlisp KL-ONE system into Franzlisp to enable it to be run on a VAX. This effort has involved Tim Finin, Richard Duncan and Hassan Ait-Kaci from the University of Pennsylvania, Judy Weiner from Temple University, Jane Barnett from Computer Corporation of America and Jim Schmolze from Bolt Beranek and Newman Inc.

The primary motivation for this project was to make a version of KL-ONE available on a PDP 11/780 VAX. A VAX Interlisp is not yet available, although one is being written and will soon be available. Currently, the only substantial Lisp for a Vax is the Berkeley FranzLisp system. As a secondary motivation, we are interested in making KL-ONE more available in general - on a variety of Lisp dialects and machines.

When we began the effort (summer 1981) we first looked at several existing inter-dialect Lisp translation systems (e.g., Interlisp's TRANSOR, SRI's MacLispify, the MIT MacLisp system developed to transport LUNAR to the Lisp Machine, and several smaller systems). None of the systems quite fit our criteria so we decided to create our own translation system. Our approach was to first build a general purpose inter-dialect Lisp translation system that is driven by transformation rules. We then developed a set of specific Interlisp to FranzLisp translation rules and an appropriate run-time support system for the resulting FranzLisp version of KL-ONE.

The current status of the project is as follows. The basic translation engine (Franzlator) has been implemented and is running smoothly. Our collection of Interlisp to FranzLisp rules, which is tailored for translating KL-ONE, numbers about forty. The run-time support environment (dubbed InterFranz) contains about 250 functions, mostly macros. In addition, a rudimentary DWIM-like facility has been developed to handle certain classes of expressions which tend to slip through the translation process.

Building a general purpose inter-dialect lisp translation system is a fairly large project in its own right and may seem to be an inefficient way to transport KL-ONE from Interlisp to FranzLisp. We have chosen to do this for several reasons. The chief ones are:

- We want a FranzLisp version of KL-ONE which tracks the current Interlisp version. Since the KL-ONE system is still evolving rapidly, this will require periodic re-translations. Thus effort spent to mechanize the translation task will pay off in the long run.

- We anticipate a desire to transport KL-ONE to other Lisp dialects such as LispMachine Lisp or Common Lisp. A properly designed inter-dialect translator will minimize the future cost of this effort.

- We expect to use the translation system to import other Interlisp systems into Franzlisp. Current candidates are RUS and PSI-KLONE. We also expect to write other sets of translation rules to use the translation system to import from other Lisp's and to export native FranzLisp programs. Thus the cost of building a general purpose translation system will be shared by other projects.

A. Translation versus Emulation

In undertaking to transport a large system such as KL-ONE from one dialect of Lisp to another there are two basic approaches: translation and emulation. Translation involves transforming the Lisp code from the initial source-dialect to the desired target-dialect. The result is a program that can be run directly in an unmodified interpreter for the target-dialect. Emulation involves reconstructing the source-dialect's environment in the target-Lisp's interpreter. Properly done, this enables the unaltered source-code to run directly. These two approaches are, of course, poles on a continuum which admit a wide range of hybrid systems.

The emulation approach, or a mixed system which is near to pure emulation, is very attractive from several points of view. An emulator tends to be easier to construct for many of the same reasons that interpreters are typically easier to construct than compilers. The emulator's task is intrinsically easier since all of the work takes place at the last moment (at run time) when all of the information is available. Once we are successful in emulating the environment, other packages of code from the source dialect can be run directly without any additional work. Still another advantage is that the source code which is run in the emulation environment is identical (more or less) with the original code. This is in contrast to a translation system which might transform readable source language code into executable, but unreadable, target language code.

In spite of these apparent advantages, we have taken a translation approach. The major reasons for this are:

- Maintaining a FranzLisp environment.

We want to maintain an environment in which any native FranzLisp code will run. Constructing a fairly complete Interlisp emulator would entail fundamental changes in the environment.

o Avoiding naming conflicts.

Many of the differences between Interlisp and Franz can be handled by adding definitions for those built-in functions which Interlisp provides but Franz does not (e.g. TCONC). In many cases, however, Franz and Interlisp use the same name for different functions. One common difference arises when the same symbol refers to two unrelated functions. The function *, for example, is a comment introducing function in Interlisp and multiplication in FranzLisp). A second class of differences arises when there is variation in the "syntax" of the function. The function MAPCAR, for example, takes it arguments in a different order in Interlisp and Franzlisp. A third class of differences involves the "semantics" of the function. An example here is LISTP function which in Interlisp returns T only for non-empty lists and in FranzLisp returns T for any list, including NIL.

o To have a stable textual Franz Lisp version of KL-ONE.

The output of the translator is a set of files which comprise a stable text-level representation of Franz Lisp KL-ONE. We believe that this makes it easier to debug, maintain and modify a large system like KL-ONE.

o Generality.

We believe that a translation approach will be easier to extend so that we can eventually produce versions of KL-ONE for other Lisp dialects. An emulation approach is more likely to depend on features of the target language will may not be present in a new candidate target language. Macros, for example, would typically be used in an emulation approach whenever possible. Some Lisp systems, such as MTS Lisp, do not support Macro functions.

ALthough we characterize Franzlator as a translation system, it is in fact a mixed system which has a significant run-time component. Most of that runtime component consists of definitions of functions which Interlisp provides but FranzLisp does not. We have chosen to define these functions as macros whenever appropriate (e.g., for simple functions like GEQ, which compares two numbers for a "greater than or equal to" relation). This has the effect of enabling us to vary the size of the run-time component by simply including a translation rule which expands calls to macros at translation time instead of run time or compile time.

II. The Translation Process

Although we have written a translator in FranzLisp, the entire translation process involves a total of four machines. The process begins on a JERICHO where the Interlisp DWIMIFY function is used to translate all of the CLISP code into standard Interlisp. The resulting dwimified files are then transferred to the BBNG machine from which they are FTPed to WHARTON-10 and finally transferred by a local networking facility to a VAX in Penn's CIS department. There the files are passed through the FRANZLATOR system to produce two sets of files. One set represents the FranzLisp version of KL-ONE and the other a collection of notes about the translation process (e.g. unrecognized functions, expressions which may require hand translation, etc.).

In translating KL-ONE, the FRANZLATOR system uses three major databases:

- o A database of Interlisp to FranzLisp translation rules.
- o A database containing information about the Interlisp system functions (e.g., function type, number of arguments, special forms).
- o A database describing functions in the InterLisp runtime environment (InterFranz).

III. Organization of the Translator

The Translator is organized as a two-pass system which is applied to a set of source-dialect files and produces a corresponding set of target-dialect files. During the first pass all of the source files are scanned to build up a database of information about the functions defined in the file. In the second pass the expressions in the source files are translated and the results written to the target files. The translation of an s-expression is driven by transformation rules applied according to an "eval-order" schedule (i.e., the arguments to a function call are translated before the call to the function itself). In addition to the transformation rules, the translator is guided by the data base of information about the functions, both the built in Interlisp functions and the user-defined ones.

An additional pass, to be done initially, may be required to perform certain character-level transformations. In translating KL-ONE from Interlisp to FranzLisp, however, we found that all of the necessary character level transformations could be done through the use of multiple readtables. The readtable used when reading the original Interlisp files ,for example, treats the character ";" as a normal alpha-numeric and "%" as an escape character.

A. The First Pass

During the first pass all of the source files are scanned to build up a database of information about the functions defined in the file. In particular, for each user-defined function we need to know how many arguments it expects and whether or not it evaluates them. The translator must know how many arguments each function expects in order to supply default values for missing arguments or to remove any extra arguments. This is important since Interlisp functions can take any number of arguments. Missing arguments are supplied as NIL arguments and extra arguments are not passed to the function. It is common practice for many Interlisp programmers to rely on this convention, especially with regards to missing arguments. An example is to write (CONS X) rather than (CONS X NIL).

The translator needs to know how each user-defined function evaluates its arguments in order to correctly translate the arguments in a call to that function. If a function parameter is not evaluated (as is the case in a Fexpr or Nlambda type function) then the translator should not translate the corresponding argument in any calls to the function. If the argument is evaluated, either by the interpreter or explicitly by a call to EVAL from within the function, the the translator must translate the argument. The problem, of course, is how to determine whether or not a function explicitly evaluates an initially un-evaluated argument.

The handling of function arguments which may or may not be evaluated is problematic in systems such as this. The proper thing to do is to examine the code to the function and try to determine whether there is an explicit call to EVAL. Franslator takes the more practical approach of assuming that the function either will or will not explicitly evaluate all of its un-evaluated arguments. The decision is controlled by the value of a global variable. A facility is supplied which allows one to directly inform the translator about a function's type, number of arguments and exactly which arguments are evaluated.

B. The Second Pass

During the second pass, each source-dialect file in the set is processed independently, resulting in a corresponding file in the target-dialect. The processing, for the most part, is simply a matter of translating each s-expression in the source file and writing the result in the target file. In addition, for each of the source files, a file containing notes about the translation is produced. Entries are made, for example, when the translator discovers a function which is not in its data base and upon encountering a function call with an improper number of arguments. In addition, any rule can add notes to this file as one of its side effects.

IV. Transformation Rules

The actual translation is done by a set of transformation rules. Each rule specifies the translation of one s-expression into one or more resultant s-expressions. In addition to the usual "pattern" and "result" parts, rules can be easily augmented with arbitrary conditions and actions.

A. The Structure of a Rule

A rule has two obligatory parts: a pattern which determines the expressions the rule applies to and a result which specifies the result of the transformation. In addition to these, a rule can have up to five optional attributes such as a test and priority. The syntax of a rule is:

```
             <rule> -> (<pattern> <result> . <attributes>)
       <attributes> -> ()  |  (<attribute> . <attributes>)
        <attribute> -> (<attribute name> <attribute value>)
   <attribute name> -> test  |  side-effect  |  priority ...
  <attribute value> -> {an s-expression}
```

Variables in the pattern are specified using a variation of the MacLisp "backquote" convention. Any symbol in the rule's pattern which is preceded by a "," is taken as a variable which can match any one s-expression. A symbol preceded by ",@" can match any number of sister s-expressions. In the result part of a transformation rule the comma and @ have a slightly different interpretation. There, s-expressions which are preceded by a "," are replaced by their values and those preceded by ",@" have their values "spliced in" in their place.

Some examples of transformation rules are shown below.

```
[r1]  (NIL nil)
[r2]  ((NLISTP ,x)  (not (dtpr ,x)))
[r3]  ((PROG1 ,@args)  (prog2 nil ,@args))
[r4]  ((MAPCAR ,list (FUNCTION ,f))
       (mapcar ',(makeMonadic f) ,list))
[r5]  ((DECLARE: ,@args)  ,(translateDeclare: ,args))
```

Rule [r1] is the simplest, mapping the symbol "NIL" into the symbol "nil". Rule [r2] introduces the use of a simple variable. The third example rule shows an application requiring a "@" variable. The rule [r4] shows an computation embedded in the result part of the rule. The second element of the result will be a list whose CAR is QUOTE and whose CADR is the result of calling the function makeMonadic with argument f. The last rule, [r5] is one in which the entire result is computed.

The optional rule attributes include TEST, SIDE-EFFECTS, PRIORITY, TYPE and REGIME. The value of a TEST attribute is an Lisp expression which must evaluate to non-NIL before the rule can be applied. The test is run after the pattern has matched so that the pattern's variables will be bound to values. An example of a rule using the TEST attribute is:

```
((PLUS ,@args1 ,x ,@args2 ,y ,@args3)        ; pattern

 (PLUS ,@args1 ,@args2 ,@args3 ,(+ x y))     ;result

 (test (and (numberp x) (numberp y))))       ;test
```

This rule causes any numeric arguments to *PLUS to be collected and summed* at translation time.

The SIDE-EFFECT attribute introduces a Lisp expression which will be evaluated whenever the rule is applied and the result has been computed. Side effect attributes are typically used to write messages about the translation into the file of translation notes or to the terminal.

The PRIORITY attribute is used to rank the rules. Whenever two rules both apply to an expression being translated, the one with higher PRIORITY is applied first. Our current Interlisp to Franzlisp translation rules do not use the priority feature.

The TYPE attribute should have as its value either splicing or replacing (which is the default rule type). A splicing rule is one in which the result is a list of expression which are to be "spliced" into the list containing the expression being translated. A splicing rule is used, for example, to transform a call to DEFINEQ into a sequence of calls to defun at the top level of the file. In a replacing rule, the result is simply replaces the original expression.

The REGIME attribute must be either cyclic or acyclic (the default case). A cyclic rule can apply more than once to the same expression whereas a acyclic rule can only be applied once. The default REGIME is acyclic. An example where a cyclic rule is appropriate is:

```
((and ,@x (and ,@y) ,@z)     ; pattern
 (and ,@x ,@y ,@z)           ; result
 (regime cyclic))
```

This rule eliminates a call to AND if it is embedded in another AND by raising its arguments.

B. Rule Representation

The translation rules are presented to the system in the form described above and are immediately "compiled" (by macro-expansion) into Lisp code. Each rule becomes a monadic function whose argument is an s-expression to be translated. If that expression matches the rule's pattern then the function will compute and return the translated form. If the expression and pattern do not match, then a special symbol indicating failure is returned. The Lisp code generated for a rule is optimized for efficiency. The pattern matching operation, for example, is "open coded" into a conjunction of primitive tests and action (e.g., EQ, EQUAL, LENGTH, SETQ).

Each rule is indexed by first assigning it to one of four classes depending on the nature of its pattern. The four classes are rules whose patterns are: (1) atomic; (2) lists with literal atoms as their first element; (3) lists with variables as their first elements; and (4) lists with lists as their first element. Rules in class two are indexed on the property list of the symbol in their CARs. The other classes are not further indexed.

C. Controlling the Translation

The translation system was designed to provide a high degree of transformational power in a simple format. A person writing a set of transformation rules may want to have greater control of the translation engine. In order to provide for such situations, the translation system makes available a number of control functions and certain relevant global variables. For example there exist functions for aborting the application of a translation rule and for prematurely ending the translation of expression without considering the application of any other rules. The rule writer has access to such values as the stack of forms undergoing translation (to allow for context sensitive rules), and the name of the current Lisp function being translated.

In addition, there are various support and debugging functions which facilitate the development of new sets of translation rules.

V. Summary, Current Status and Future Directions

This section has reported on the development of a general inter-dialect Lisp translation system and its application to the task of translating the Interlisp implementation of KL-ONE into FranzLisp. The translation system is running smoothly and fairly efficiently. The current set of translation rules and run-time support functions appear to cover all of the basic facilities needed by KL-ONE. We are currently in a cycle in which a translation of KL-ONE is made and then run to discover bugs in the translation rules or run-time support system.

Some future work will be directed towards experimenting with extensions to translation system to allow for more flexibility, power and/or efficiency. Other work will involve broadening the set of interlisp to Franzlisp translation rules to handle constructions not required in translating KL-ONE and to handle CLISP code. A third direction is the writing of rules for translating between other pairs of Lisp dialects. A set of Interlisp to CommonLisp rules might be very useful, for example.

Towards a calculus of structural descriptions (or, how to do away with arbitrary preemption in specialization hierarchies)

Michael Freeman, Chris J. Tomlinson (co-authors),
Donald P. McKay, Lynette Hirschman,
David P. Oster, Karl O. Puder.

Information Modeling and Management
R&D/FSSG
Burroughs Corporation

(Presented during the main conference)

This paper is based on the following premises:

1. Structural Descriptions (SD's) can be specified as sets of logical clauses (propositions, assertions);

2. A is a superC of B only if given the SD of B, one can prove or derive that of A (viz, B $\vdash$ A); in other words, the SD of B represents a theory that is a specialization of the theory represented in the SD of A.

There are two classical manners in which theories can be specialized or strengthened: 1) through substitution of terms, 2) through addition of axioms. For example, if an "arch" is a kind of "structure", then by (1) all the axioms and theorems of "structure" become part of the "arch" theory through substitution of the term "arch" for that of "structure" in all axioms and theorems in which the latter is mentioned. By (2), we may add axioms such as "If x is a component of an arch, then x is either a lintel or an upright."

There is a third manner in which one might wish to specialize a theory, however. This is one familiar to people working with associative network types of representation and involves specializing individual axioms other than through substitution. For instance, starting from "the height of one upright of an arch is greater than or equal to the height of the other upright of the arch", one might wish to specialize to "the height of one upright of a slanted arch is greater than the height of the other upright of the arch."

Now, it turns out that if one places the propositional connectives on a generalization/specialization hierarchy, one gets the lattice shown in Figure 1.

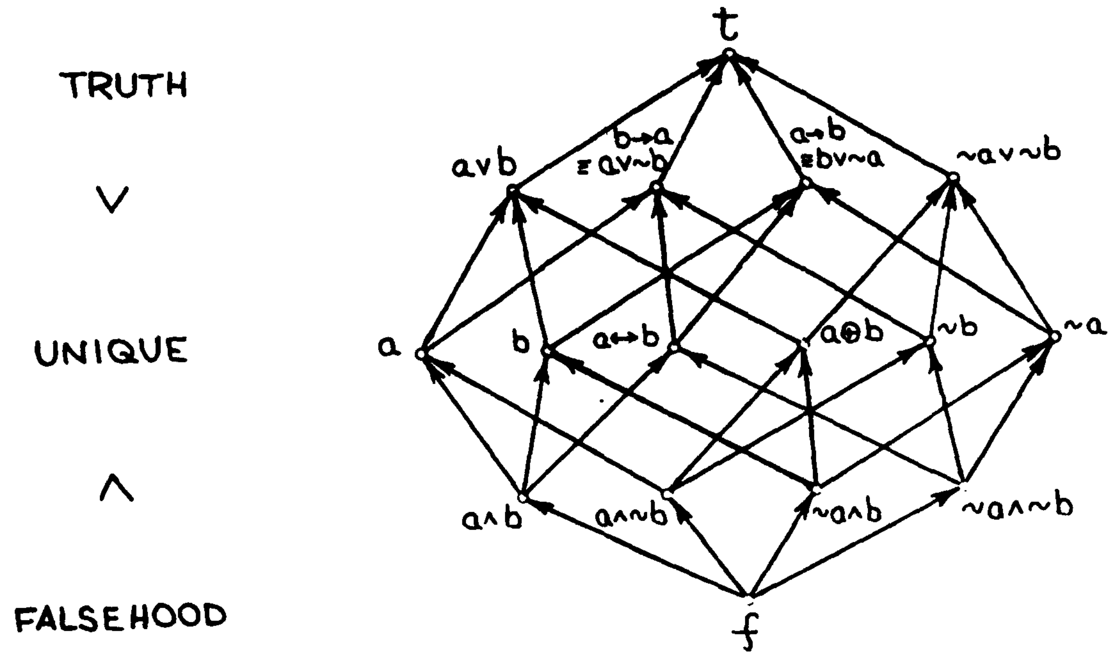


FIG. 1. GENERALIZATION/SPECIALIZATION LATTICE

As expected, one finds here that unique characterization specializes or strengthens disjunction, and is itself specialized or strengthened by conjunction. What happens, however, if one specializes a term in a particular disjunction or conjunction? Is the resulting disjunction or conjunction itself a specialization of the original one? This can be shown to indeed be the case, provided one is dealing with non-negated predicates. In the case of negated predicates, however, just the opposite obtains. In fact, this holds true even at the level of unique characterization (e.g., NOT(STRUCTURE) is a specialization of NOT(ARCH) rather than the other way around.) This has obvious consequences for the specialization of (material) implications. In particular, by specializing the consequent of a conditional, one obtains a more specialized version of the implication, but in specializing the antecedent, what one ends up with is a weaker or more general version of the implication. Stated more formally:

```
[1] (A' --> B')  ===>  (A --> B)  iff
     [(A = A') ^ (B' ===> B)]  V  [(A ===> A') ^ (B' = B)]
       V  [(A ===> A') ^ (B' ===> B)]
```

where "A ===> A'" is to be read as "A is a specialization of A'" (i.e., logically entails it).

To get an intuitive feel for the validity of this meta-principle, consider the following pair of propositions:

```
[2]   NOT(Detected(x))  -->  Safe(x)

[3]   NOT(DetectedByReconnaissanceUnit(x))
        -->  SafeFromReconnaissanceGeneratedAttacks(x)
```

For most people [3] is generally accepted as a specialization of [2], even though its antecedent is more general than that of [2].

Where there is no linguistic negation in the antecedent, however, one's intuitions seem much less clear, as the following modifications to [2] and [3] may help to demonstrate:

```
[2']  Detected(x)  -->  InJeopardy(x)

[3']  DetectedByReconnaissanceUnit(x) --> InJeopardy(x)
```

Since the antecedent in [3'] is a specialization of the one in [2'], and the two consequents are the same, then according to [1] we should find [2'] to be a specialization of [3']. At first glance, this seems counterintuitive. The problem here is that [3'] doesn't distinguish between cases in which something is detected, albeit not by a reconnaissance unit, and those in which it is simply not detected at all. Given the truth table for material implication, [3'] would be considered true in both these cases, whereas what one normally intends is that complete lack of detection would in fact remove one from being in jeopardy (i.e., [3'] should be false in this case). Pursuing this line of reasoning, however, one sees that [2'] can also be regarded as suffering from exactly this same type of underspecification, i.e. if one is NOT detected, then one is not in jeopardy (at least not in jeopardy from having been detected). Although it may seem that what's at fault here is the peculiar nature of material implication itself, there's another way of looking at the problem which not only gets around the difficulties raised so far, but also seems to capture a basic insight in doing so. What we propose is simply that one specify along with the "primary" implication a "secondary" one that takes into consideration those cases arising from a failure of the antecedent in the first. This gives rise to conjoined implications, much along the lines of "if-then-else" statements. In forming the antecedent for the second conjunct, however, one does not merely take the negation of the first conjunct's antecedent; rather one takes the relative complement of the latter with respect to the antecedent of the more general implication one wishes to specialize. And therein lies the insight. Thus, if we transform [3'] in line with what has just been suggested, we get:

```
[3"] {[DetectedByReconnaissanceUnit(x)  -->  InJeopardy(x)]}
       ^ {[(Detected(x) ^ NOT(DetectedByReconnaissanceUnit(x))]
        --> [InJeopardy(x) ^ NOT(InJeopardyFromDetectionByRecon(x))]}
```

Schematically what we are proposing then is to consider the following as a proper specialization of the implication "A --> B":

```
[4]   [A' --> B'] ^ [(A ^ ~A') --> (B ^ ~B')],
```

where A'===> A and B' ===> B. A formally equivalent and perhaps somewhat more perspicuous way of representing this is the following:

```
[4']  A --> [(A' --> B') ^ (~A' --> (B ^ ~B'))]
```

Since "A --> B" can be derived in a straight-forward manner from [4'], the latter also constitutes a valid specialization of the former from the purely formal point of view set forth in our second premise. Note that even though replacing the second conjunct in [4] by "A --> B" would also allow one to make this derivation trivially, one would not have a formally equivalent conjunction. To see this, consider the case where all the predicates except B' are true. The proposed "simplification" of [4] would yield true, whereas [4] itself would not.

We should point out that it is possible to achieve the same results with a slightly weaker version of [4] or [4'], namely one in which the final consequent is simply some arbitrary specialization of B, say B", rather than (B ^ ~B'). What we would like to propose is that when this is the case, then it is necessary to express the specialization in a fully explicit form, e.g.:

[4"] A --> [(A' --> B') ^ (~A' --> B")]

Otherwise, one need specify simply "A' --> B'" as a specialization of "A --> B", and the system then automatically fleshes this out in accordance with the schema given in [4] or [4'] (provided, of course, that A' ===> A and B' ===> B).

Given our view of what constitutes a proper specialization of an implication, an obvious consequence is that as one proceeds down an implication hierarchy, specializations become more and more embedded within the universe of some most general antecedent. This still leaves open the question of how the top-most implication is to be treated. In particular, what is the universe with respect to which one specifies the relative complement for the antecedent of the "else" conjunct? This, of course, is precisely the same problem that needs to be addressed in any hierarchy involving negation. As long as we can relativize negative concepts with respect to our most abstract "Thing" concept, then we are all right. But as soon as we wish to allow in "NOT(Thing)", we are in trouble. Let us therefore stipulate that whatever resides at the top of a hierarchy cannot be specialized through negation. Applying this principle to a predicate hierarchy enables us to guarantee that there will always be some universe with respect to which we can take the relative complement of an antecedent, provided that we never use that predicate itself in an antecedent. Note that in this view of things, material implication loses much of its peculiarity and in fact can be taken as an abbreviation for those cases in which the consequent of the "else" conjunct of our schema is simply "true".

Another area in which the type of theory specialization discussed above is directly applicable concerns our work on Augmented Event Transition Networks (AETNs). An AETN is a kind of event grammar that forms part of the specification of a conceptual entity for us. It serves as the basis for modeling the role-dependent aspect of the entity, specifically: the "expected" behavior of participant role-players in events that have extended time-horizons (e.g., contractual obligations and rights of socio-legal entities such as LANDLORD or EMPLOYEE), and the possible transformations which physical entities can undergo, relative to particular functional perspectives, without destroying their "objecthood". As formal objects, AETNs can be represented by sets of logical clauses or propositions incorporating special operators and connectives for dealing with such notions as temporal realization, precedence, possible or eventual succession, etc. The latter type of connectives, given a suitable axiomatization, can be arranged on a generalization/specialization hierarchy in much the same way as the logical connectives examined earlier. Of particular interest are the two connectives for "eventually (in every future)" and "possibly (in some future)", which we will represent by "~~>" and "``>" respectively. Thus "p ~~> q" is to be read as "p eventually leads to q", and "p ``> q" as "p may (possibly) lead to q" (i.e., there exists at least one possible future in which p does in fact lead to q). Clearly for p to eventually lead to q, it must ipso facto be possible for p to lead to q. Thus "eventually" should be a specialization of "possibly", and indeed it can be proved formally that:

[5] (p ~~> q) --> (p ``> q)

In giving a logical specification for AETNs, two types of concern need to be kept in mind. The first has to do with the inferences that can be drawn from an initial set of axioms, in order to make explicit the fully expanded version (or "theory") of an AETN. The second has to do with the inheritance rules for AETNs, guiding the construction of complete AETNs out of "fragments" distributed throughout the generalization/specialization hierarchy.

As an informal example of the first concern, consider the following two axioms of an AETN for a "marriageable" person:

[6.1] Birth eventually leads to death.

[6.2] Birth may possibly lead to marriage (provided death doesn't intervene).

Clearly, one of the things one should be able to conclude from this is that if marriage indeed does take place, then eventually death still is going to take place, i.e., marriage eventually also leads to death (not necessarily in any causal sense!). Furthermore, one should also be able to conclude that it is not possible for death to lead to marriage.

In order to illustrate the second concern, let [6.1] represent the specification of one AETN and [6.2] the specification of another one. By making [6.2] a specialization of [6.1], we in essence wind up with the AETN discussed in the previous example.

Before listing a somewhat more formal version of the two types of deductions illustrated above, we will introduce one last bit of notation in the interests of simplifying our representation:

```
[7]   p |~~> q   ::=  p ~~> q ^ Vr(NOT(q ``> r))
```

The following then are examples of deductions associated with expanding an initial axiom set characterizing an AETN:

```
[8] [(p |~~> q) ^ (p ``> r)]  -->  [(r ~~> q) ^ NOT(q ``> r)]
```

(This is simply a restatement of the example connected with [6.1-2]. In all subsequent formulations, we will use the "|~~>" construct in the consequents also, thus eliminating the need for any "NOT" conjuncts.)

```
[9] [(p |~~> q) ^ (p ``> r) ^ (p ``> s)]
        -->  [(r |~~> q) ^ (s |~~> q)]

[10] [(p |~~> q) ^ (p ``> r) ^ (p ``> s) ^ (r |~~> s)]
        -->  [(r |~~> q) ^ (s |~~> q)]
```

Associated with the specialization of AETNs are a number of deductions such as the following:

```
[11] Let net A = p |~~> q, net B = p ``> r, and B ===> A.
        The full expansion of B =
               (p |~~> q) ^ (p ``> r) ^ (r |~~> q).
```

(This is a restatement of the example wherein [6.2] ===> [6.1].)

```
[12] [(A = (p |~~> q))  ^  (A <=== B, wherein q' ===> q)]
        -->  [B = p |~~> q']

[13] [(A = (p |~~> q)) ^ (B ===> A, wherein p' ===> p)]
```

```
        --> [(B = p' |~~> q)]

[14] [(A = (p |~~> q) ^ (p ``> r)) ^ (B = (r |~~> s) ^ (r ``> t))
        ^  (C ===> (A ^ B))]
        -->  [C = (p |~~> q) ^ (p ``> r) ^ (r |~~> q) ^ (r |~~> s)
         ^ (r ``> t) ^ (t |~~> s) ^ (s |~~> q) ^ (t |~~> q)]

[15] [(A = (p |~~> q) ^ (p ``> r))  ^  (B = (p |~~> s) ^ (p ``> r))
        ^  (C = (A ^ B))]
        -->  [C = (p |~~> q) ^ (p ``> r) ^ (r |~~> q) ^ (r |~~> s)
              ^ (p ~~> s) ^ (q ~~> s)].
```

(See Figures 2 and 3 for a graphical representation of [14] and [15].)

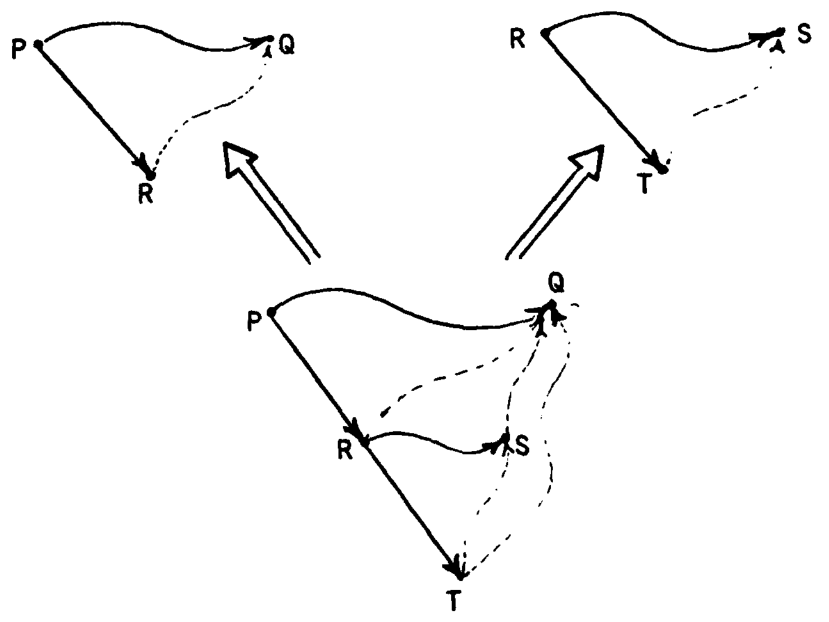


FIG. 2.

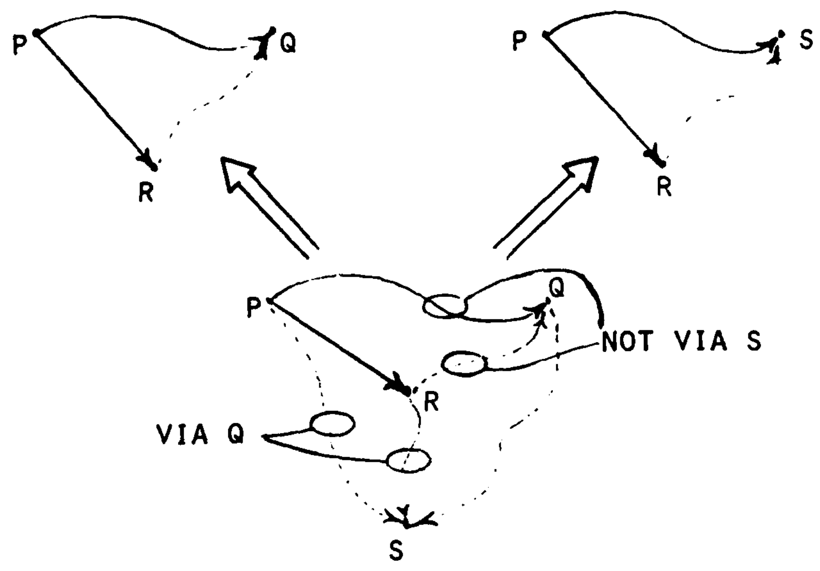


FIG. 3.

A Knowledge Representation Model of Prototype Theory

Benjamin Cohen

Cognitive and Instructional Sciences Group
Xerox Palo Alto Research Center
Palo Alto, California

(Presented during the main conference)

I'm currently using KL-ONE, or at least something that bears a family resemblance to KL-ONE, as a modelling tool in cognitive science. I want to model a recent theory of concepts known as prototype theory developed over the last ten years by the psychologist Eleanor Rosch up at Berkeley and a number of others in cognitive science. There are a variety of motivations for modelling prototype theory. My own motivation is to reply to a recent critique of prototype theory developed by Dan Osherson and Ed Smith, two psychologists who think there's something fundamentally incoherent about prototype theory. They base their critique on a fuzzy set model and I want to use knowledge representation as an alternative to the fuzzy set model which is not subject to their criticism.

1. OVERVIEW OF PROTOTYPE THEORY

Prototype theory in psychology is a well-established body of experimental and theoretical doctrine. The core is the view that concept instantiation is not all-or-none but more-or-less, not a matter of satisfying a definition but a matter of typicality or resemblance to a prototype. For example, if I ask you what is a typical bird you are likely to say robin, than either chicken or penguin. Facts such as these show up in a variety of experimental paradigms as well as in natural language.

The traditional definitional picture of concepts which goes back to Aristotle, represents concepts by a statement of necessary and sufficient conditions--the paradigm being the definition of bachelor as unmarried male. To see the difficulties with the definitional view you don't need to go into the lab and do reaction time experiments. A quicker approach is the following thought experiment. Choose your favorite natural kind term from the dictionary. If we put the dictionary definition of horse into the form of necessary and sufficient conditions we get something like the following.

For all x, x is a horse if x has 4 legs & x eats grass x & x is an animal & x is used for riding.

With the possible exception of animal, none of these conditions are individually necessary let alone jointly sufficient for horsehood. A 3-legged horse that hated grass and was never used for riding is an atypical horse, but a horse no less. What about animal? I claim there's no contradiction involved in the notion of a robot horse--it's just atypical. The same holds for female bachelors. A more accessible example is the definition of father as male parent. The typical father is a biological parent, but there are fathers who are not biological parents. Again there's a grading off with the prototype father being a biological parent.

What prototype theory provides is an empirically motivated alternative to the definitional account. Now there may be other perfectly legitimate motivations for choosing a terminological or definitional approach to concepts. So my position need not be regarded as in conflict with the current definitional approach to KL-ONE.

2. THE FUZZY SET MODEL

O&S embed the fuzzy set model in a referential theory of concepts as classes or extensions rather than a theory of concepts as mental representations. The concept BIRD for example is a quadruple <B, b, c, d> where B is the class of birds, b is the prototype bird, c measures "birdiness" and d is a distance metric on pairs of birds. Using this scheme the core empirical findings of prototype theory can be captured by constraining c to vary inversely with d. But as is typical of the extensional approach notice how little knowledge of birds is encoded. There is no syntactic or lexical information and no mention is made about <u>bird's</u> relation to other concepts.

The thinness of this extensional account can be thickened to a structured fuzz by using roles and inheritance from standard KL-ONE. So far no problem. The question is how can we represent typicality in a KL-ONE-ish setting? One suggestion is to put explicit real-values on links to indicate semantic distance. This has an appealing vividness and I actually have proposed this but have since thought better of it for it seems a very strong assumption to make, particularly when outside of experimental contexts it is never clear what a given number means. So instead I use partial orderings, partial orderings of instances, roles-values and sub-concepts. A concept is regarded not as a definitional entity but as set of instances partially ordered by their typicality or degree of resemblance to a distinguished instance called the prototype.

What is the real problem with this theory of partial orderings? It's not absence of formal detail, but the lack of a <u>theory of resemblance</u>. This is a matter for empirical and theoretical investigation that psychologists like Tversky have only recently begun to seriously investigate.

3. THE PROBLEM OF COMPLEX CONCEPTS

O&S state the following compositional criterion of adequacy for any theory of concepts. Given an arbitrary mode of combination *, a theory of concepts should specify the representation for X*Y solely in terms of the representation for X and Y. Thus for conjunctive concepts, the fuzzy set model should tell us how to determine the quadruple for the conjunctive concept X&Y solely on the basis of the quadruples for X and Y.

The min rule is the fuzzy set solution for conjunctive concepts.

X&Y(x) = min(X(x), Y(x))

It says that something's X&Y-ness is the min of its X-ness and its Y-ness. An obvious consequence of this rule that the typicality of an instance of a complex concept is never greater than that for the constituents. Yet as O&S point out this is very counter-intuitive. My pet guppy is a more typical pet fish than either a pet or a fish.

We could try to improve the min rule in the fuzzy framework but it will still fail for the great mass of non-intersective NPs like large mosquito, olive oil, motor oil, toy factory, kitchen knife. There is much more to said about the problem of complex concepts. The lesson I want to draw is the inadequacy of know-nothing extensional semantics to account for our competence. Whatever this competence involves it is not a matter of combining fuzzy extensions using simple rules of combination. At the very least extensive knowledge of prototypes and rules for constructing prototypes for complex concepts is required.

4. A KNOWLEDGE REPRESENTATION APPROACH

In general in attempting to understand anything there may either be too few representations--the nonsense case like green idea--or too many--the ambiguous case like aluminum soup pot which may either be a soup pot made of aluminum or a pot full of Campbell's aluminum soup.

The basic model of complex concepts I propose is a pretty straightforward application of KL-ONE which introduces the notion of a mediating role. The reason 'green idea' is anomalous is the absence of a mediating color role on the concept idea which takes green as value. Ideas don't have color. The reason green apple makes sense is because apples inherit a color slot (say) from physical object which takes green as a value. This value is understood to restrict the mediating role. The reason aluminum soup pot is ambiguous is that there are two mediating roles, a made-of role on pot and one on soup.

This theory divides KL-ONE nodes into two classes: compositional and non-compositional. Compositional nodes are nodes that have structure and are used to represent prototypes for complex concepts like aluminum soup pot where specialization is matter of restricting values of mediating roles. We use '|' (read slash) for composite nodes and write pot|soup|aluminum where the mediating roles are made-for on pot which has soup as a restricting value and made-of on pot (perhaps inherited from a super-concept) which has aluminum as a restricting value. To summarize here's a rough definition of mediating role. A role R is a mediating role for concept C and modifier m in network K.

1. R is a role on the prototype instance of C and m satisfies the value restriction for R or

2. R is an inherited role for C in K and m satisfies the value restriction for R.

In addition to providing an alternative to the min rule and its ilk, I believe this theory of complex concepts can be extended to handle the classical problem of default conflicts in multiple inheritance schemes. Conflicts like the case of Clyde, who is both elephant--hence gray if he is typical--and an albino--hence white if he is typical--may be resolved by finding the complex prototype which is most typical. In this case albino elephant in which elephant is the head and albino the modifier wins over elephant albino since the latter does not resolve the color ambiguity. Assuming the knowledge-base is not simply contradictory or ambiguous the conflict can be resolved by the principle of typicality: Choose the most typical interpretation. Here albino elephant is more typical than elephant albino since the latter does not resolve the color ambiguity.

A KL-ONE Classifier

Tom Lipkis

USC/Information Sciences Institute

(Presented during the main conference)

1. THE NEED FOR CLASSIFICATION

Taxonomies are one of the most natural and useful ways to organize descriptive terms in a knowledge base. They are easy for people to understand, and they facilitate many common operations such as determining whether one term is an instance of another, and finding all instances of a generic term. While static taxonomies can be constructed manually, maintaining dynamic taxonomies requires the ability to automatically classify new terms -- that is, to procedurally determine the taxonomic relationship between a new term and existing terms, and then to incorporate that relationship into the knowledge base. Many AI programs use taxonomic knowledge bases (semantic networks) to model changing domains, and therefore require the automatic classification of new knowledge as it is obtained. Automatic classification also provides a means of enforcing network semantics and checking consistency of descriptions, and is therefore a superior alternative to manual construction in the implementation of static taxonomies.

This note describes a classifier developed for the KL-ONE representation language (see [1], [2]). Some examples of the use of classification in the Consul system (see [3]) are first presented, followed by a brief discussion of the import of network semantics on classification. The semantics of KL-ONE as it relates to classification is then described, and finally an outline is given of the classification algorithm itself.

2. USES OF CLASSIFICATION IN CONSUL

The key to Consul's ability to provide users with a cooperative interface to interactive services is the large body of knowledge it has about those services. Its knowledge base consists of both service-independent and service-dependent knowledge. The service-independent knowledge models interactive computer systems in general, describing a taxonomy of concepts common to interactive services (e.g., display operations, delete operations, files and messages). These service-independent concepts are instantiated by service-dependent concepts that describe the actual objects and functions available in a particular service. Consul uses this knowledge base to map descriptions of user requirements (parsed user requests) into actions to be performed by the system (either service execution or Consul explanation of service capabilities).

The situations in which the classifier is invoked fall into two categories: network building and mapping.

- Network Building: Network building in Consul occurs in two different contexts: building and maintenance of the initial service-independent knowledge base by Consul designers, and automatic acquisition of new service-dependent knowledge by Consul's acquisition component (see [4]). The classifier is used in network design to make sure that the piece-by-piece construction of the network produces a valid KL-ONE structure that can be used for later information retrieval. The classifier also insures that all changes to network structures are reflected in appropriate changes in the network taxonomy. New knowledge is obtained through interaction between individual service builders (having no knowledge of KL-ONE) and Consul's acquisition component. As a service builder defines or changes service objects and functions, the acquisition component calls on the classifier to insert them into the knowledge base and check their validity. Information about the classification of these objects and functions is then presented to the service builder for verification.

- Mapping: Mapping in Consul is a process of redescribing structures by applying inference rules of the form "instances of X can be redescribed as instances of Y," where X and Y are existing structures in the knowledge base. When a rule is applied (because a description of interest is found to instantiate its antecedent (X)), a new structure is built which instantiates the rule consequent (Y). This new structure is then classified, perhaps instantiating another rule antecedent.

3. CLASSIFICATION AND NETWORK SEMANTICS[1]

Most current semantic network formalisms represent taxonomies through some kind of "is-a" link and some notion of inheritance of properties down is-

[1]See [5] and especially [6] for more detailed discussions of these issues.

a links. Most, however, are unclear as to the semantics of a term, or description. Many treat descriptions as prototypes, expressing default conditions which can be violated. For example, walruses have two tusks, but we may know that Herbert the walrus has only one tusk. Herbert certainly is a walrus, so we merely cancel the inherited "number of tusks" property.

Defaults of this sort are extremely useful to AI programs, but their use seriously affects the expressive power of a representation language. The fact that properties of a description can be cancelled means that these properties are non-definitional. Without the ability to analytically define terms, one cannot create composites (e.g., the concept of a four sided polygon), and so every term in the language is primitive. With only primitive terms, the system cannot tell whether one term is a specialization of another, so automatic classification of new terms is impossible.

In KL-ONE, cancellation of properties is not allowed, and descriptions <u>are</u> definitional. Thus automatic classification based on the structure of terms is possible. Furthermore, KL-ONE distinguishes between the operations (links) used to define concepts as formal objects in the network, and the domain-dependent relationships between the elements of that domain. The existence of a separate "epistemological" level of representation makes it possible to write general, domain-independent routines for constructing and maintaining networks. In KL-ONE, the fundamental network creation and maintenance process <u>is</u> classification: the determination of the taxonomic relationship between objects, and the incorporation of this relationship into the network.

4. SEMANTICS OF A KL-ONE TAXONOMY

A KL-ONE description is in the "right place" in the taxonomy if it specializes or instantiates the most specific description(s) which subsume it, and is specialized or instantiated by the most general description(s) which it subsumes. This section briefly describes the structure of KL-ONE descriptions, then defines the semantics of the subsumption relationship in terms of that structure.

4.1. KL-ONE Structure

A KL-ONE network consists of two distinct parts: a taxonomy of intensional descriptions and an extensional database of objects ("nexuses") to which descriptions can be attached. The classifier deals only with the intensional taxonomy[2]. Descriptions are represented by "concepts," and the specialization (is-a) link is called a "superconcept cable." Concepts have attributes which are represented by "rolesets" (often referred to simply as "roles"). Rolesets consist of facets which define constraints on the set of values that may fill them. The cardinality of the fillers is specified by a range called the "number restriction," and the type of filler by a "value description," which is a concept. For example the concept Quadruped has a limb role whose number restriction is four, and whose value restriction is the concept MammalLimb. This structure is shown in the top concept in Figure 1.

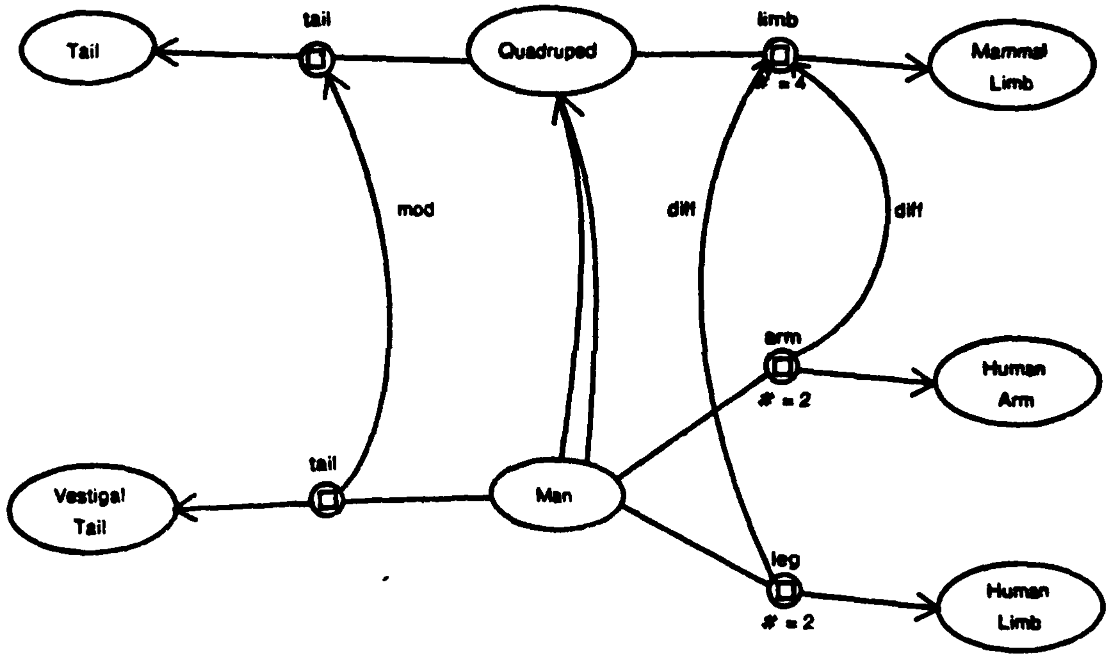


FIG. 1. KL-ONE STRUCTURE

The roles of a concept are inherited by all its subconcepts unless they are explicitly modified. Modification indicates a restriction of the

[2]The realizer deals with the extensional, see [7]

attribute of the superconcept and is represented by a "mod" wire from the role of the subconcept to the role of the superconcept. A role can also be differentiated, which indicates a decomposition of the fillers of the role into distinguished subsets. Examples of these are shown in Figure 1, where the tail role of of a Man is restricted to be a VestigalTail, and the limbs of a Quadruped are decomposed into Man's arms and legs. Roles of a superconcept which are differentiated but not modified are still inherited, as the differentiators are not necessarily exhaustive. Thus Man still inherits a limb role which is differentiated by arm and leg. Differentiation of roles within a single concept is also permitted.

Constraints on the relationships between fillers of roles are specified by "role set relations" (RSRs). In the most general kind of RSR, the condition is specified by a "para-individual concept," an instantiation of a generic concept which is parameterized to refer to parts of another description. The roles of a para-individual are called "coref roles" because their fillers are co-referential with the fillers of the generic roles they point to. An example is shown in Figure 2, where the uprights of an Arch are constrained to support the lintel.

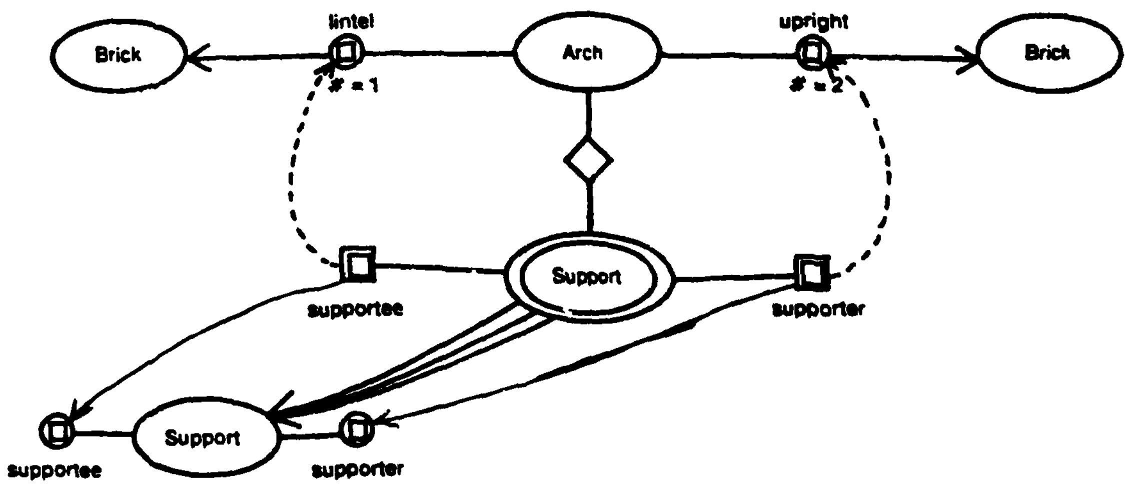


FIG. 2. A ROLE SET RELATION

"Role value maps" (RVMs) are a restricted form of RSR in which a built-in condition is specified in place of a para-individual. RVMs always have two coref roles, and specify either that the set of fillers of one role is equal to the set of fillers of another, or that the fillers of one are a subset of the fillers of another.

In order to accommodate primitive concepts such as Dog or Red which cannot be criterially defined, concepts can be marked as "natural kinds," which means that their definitions specify necessary but not sufficient conditions. It is not possible to recognize instances of natural kinds by their properties.

4.2 The Right Place

Because a concept in a KL-ONE network inherits properties from its superconcepts, a superconcept cable can have one of two distinct meanings (which are not distinguished in KL-ONE). If all of the properties which the subconcept inherits from the superconcept are also represented explicitly on the subconcept, then the superconcept cable does not affect the definition of the subconcept, and merely represents a relationship that happens to exist between the two concepts. If, on the other hand, there are some properties which the subconcept has only because it inherits them from the superconcept, then the superconcept cable is definitional; the subsumption relationship depends on the presence of the cable. Definitional superconcept cables are created by processes which build new concept structures. Putting each new structure in the right place requires finding all of the happenstance subsumption relationships between it and existing structures, and creating superconcept cables to represent these relationships.

In terms of KL-ONE structure, a concept Super subsumes a concept Sub if:

- Each role of Super is modified by a role of Sub.
- The value description of each role of Super subsumes the value description of the corresponding role of Sub.
- Sub has a subset relationship (differentiation) between any roles corresponding to roles of Super which have a subset relationship.
- The number restriction of each role of Super subsumes the number restriction of the corresponding role of Sub.
- The constraint specified by each role set relation of Super is met by Sub.
- All primitive (natural kind) concepts that are ancestors of Super are also ancestors of Sub.

A concept subsumes itself and all of its specializers; a role set subsumes itself and all of role sets that modify or differentiate it; a role set relation subsumes itself and all role set relations whose components are subsumed by its components. These relationships are shown in Figure 3.

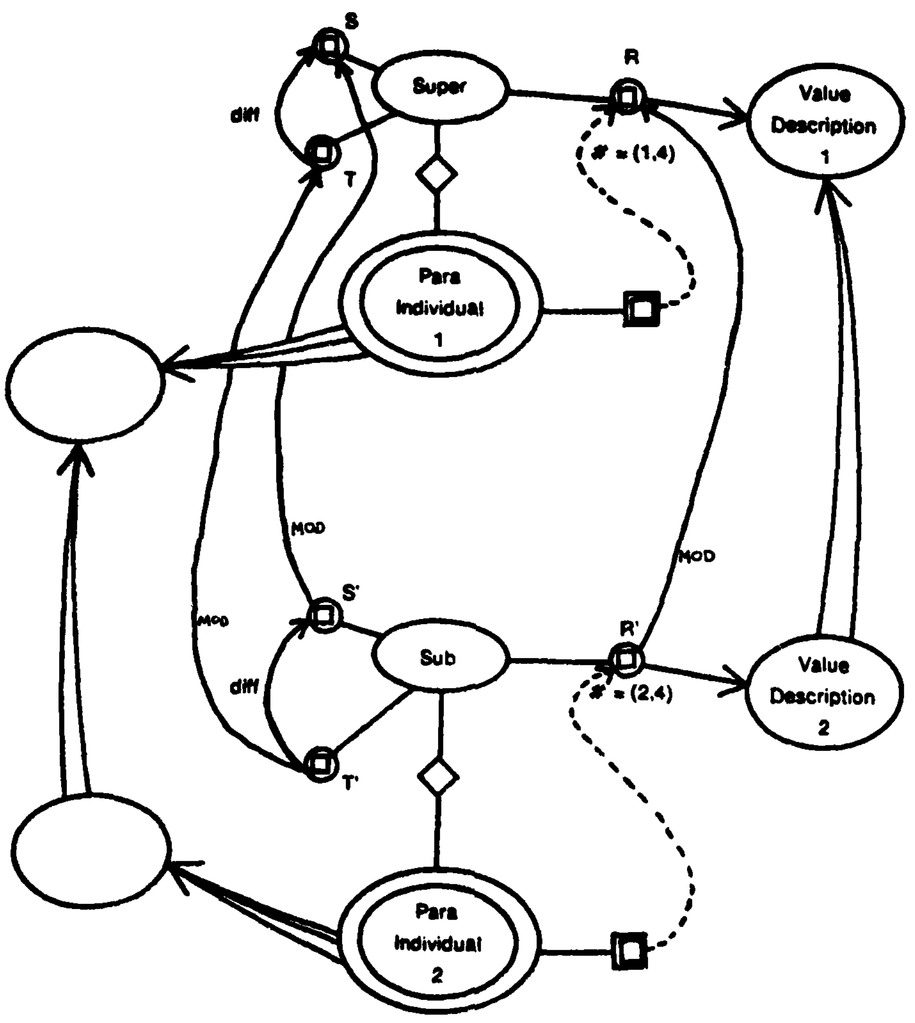


FIG. 3. SUBSUMPTION RELATIONSHIPS

4.3. Issues in Determining Subsumption

While the subsumption criteria are defined explicitly by KL-ONE, there are some open questions involving the type and extent of reasoning that is appropriate for the classifier to perform. One question regards the amount of effort the classifier should expend to determine whether some subsumption relationship holds. For example, in order to make use of concepts representing integers, one might encode an axiomatization of the integers into KL-ONE structure, and then expect the classifier to be able to determine that the concept Integer subsumes the concept 439. With an appropriate reasoning ability the classifier could do this, but the time involved would likely make it impractical. While it seems useful for the classifier to be able to perform reasoning of this nature, it is clearly necessary to limit the amount of resources that it can consume.

Another question is whether the classifier should use knowledge which is not explicitly represented in the network. One alternative to axiomatizing the integers in KL-ONE is to encode a certain amount of number theory in the classifier itself. It would then be quite easy to determine that Integer subsumes 439 (though probably not to determine that EvenInteger subsumes The57thDigitInTheDecimalExpansionOfPI), but this would violate a principle of KL-ONE; namely that concepts are defined purely by their structure, and carry no "hidden" meaning.

These same issues arise in the determination of whether an RSR constraint on one concept is met by another concept. If that other concept has an RSR specifying the same constraint, then it can be trivially determined that the constraint is met, as it is stated explicitly. If, however, the second concept specifies a stronger constraint, some amount of reasoning may be necessary to discover that fact. Another possibility is that the second concept specifies no constraint explicitly but still satisfies the required constraint by virtue of its structure. For example, an RSR which specifies a LessThan relationship between the fillers of two roles is satisfied in a concept in which the value descriptions of the two roles are 6 and 7, respectively. Again, it would be useful for the classifier to make this determination, but here, in addition to the two issues previously mentioned, there is a third issue involved: the distinction between intensional and extensional knowledge. Role set relations express constraints between the fillers of roles, and while it is certainly inappropriate for the classifier to use extensional knowledge about the actual role fillers in any instance, in some cases (such as this example) there is sufficient information about what those fillers can be to determine that the constraint must be true in any extension.

Unless the decision is made to allow the classifier to consume an arbitrary amount of time reasoning, there can be no guarantee that it will find all the subsumption relationships that hold in the network. Thus, as a practical matter, the classification process is incomplete, and the taxonomy is only a partial representation of the set of subsumption relationships.

5. THE ALGORITHM

The job of the classification algorithm is to determine the right place in the network for each incoming description, and to establish the description in that place. This is done by searching the existing network to find the most specific concepts that subsume the new description, and the most general concepts that it subsumes, and then making these subsumption relationships explicit.

New descriptions are always built as refinements, elaborations, or combinations of existing descriptions in the network. Every new description therefore has an initial "place" in the network by virtue of this construction process. However, this may not be the "right place," since the description has not yet been formally classified. Each new description must therefore be classified -- made part of the taxonomy -- after it is built. Once a concept has been established as part of the taxonomy, the classifier assumes it will not be changed or deleted, as the effects of such changes on other concepts in the network are, in general, unpredictable.

A description is normally classified by first classifying the concepts which make up its subparts (role value descriptions), then the description head concept itself. The reason for this is illustrated by Figure 4. In order to determine whether SendOperation1 is subsumed by SendOperation, it must be determined whether Message subsumes Message1. For the purpose of classifying SendOperation1 it would be sufficient to make this test, establish Message1 as a subconcept of Message, and go on. However, Message1 is an incoming description in its own right and so its right place must also be found. In this case Message1 is also subsumed by TypedMessage, but this would not be discovered by the above top down procedure. Message1 must therefore be classified independently of the classification of SendOperation1. This can be accomplished by classifying descriptions in a bottom up manner.

However, in the event that part of the description is cyclic (as in Figure 5), classification of the cyclic part cannot proceed bottom up. But, since each concept in the cycle has as one of its subparts the entire cycle, the problem of classifying Message1 shown in Figure 4 cannot arise. Thus, there is no need for independent classification of the subparts. Testing the subsumption relationship between any concept in the cycle and an existing concept still requires knowing the relationship between the concepts' subparts. Since one of these subparts is the entire cycle, all concepts in a cycle must be classified in parallel. This is performed by assuming that the

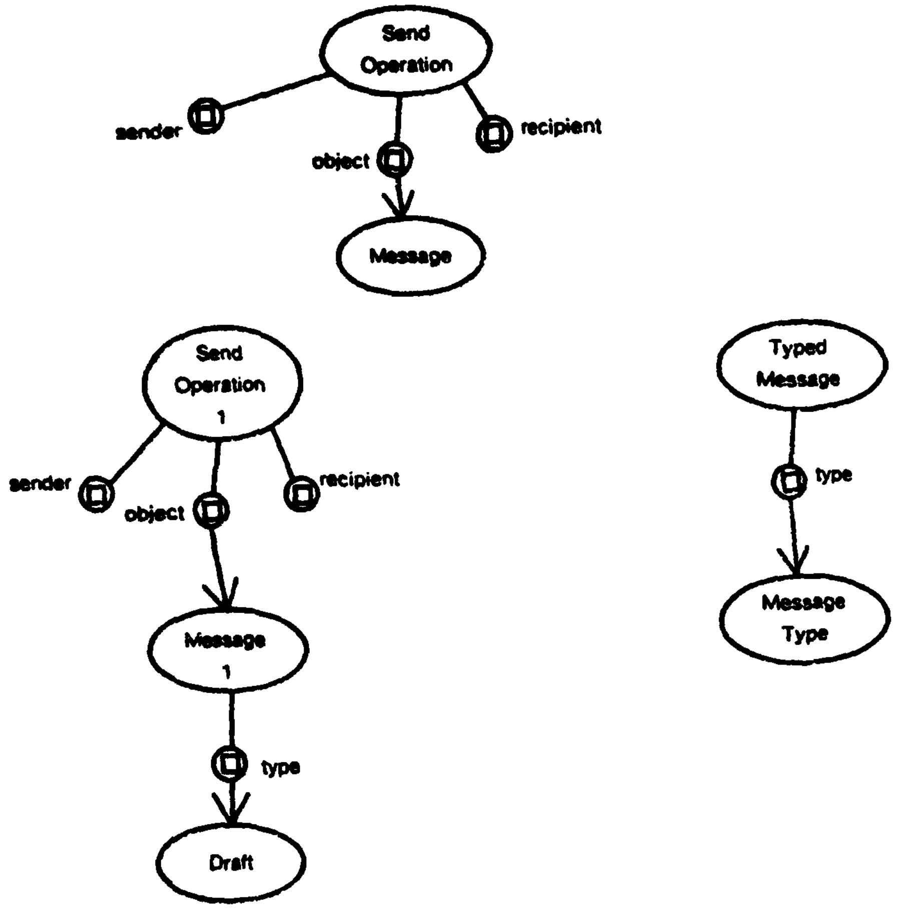


FIG. 4. CLASSIFICATION OF SUBPARTS

necessary subsumption relationships exist for each subpart, then checking to see if that is actually the case. This parallel classification is a generalization of the algorithm presented in the remainder of this section.

The overall classification problem has now been reduced to the problem of classifying a concept, all of whose subparts have been classified. For each such concept, two sets of concepts must be found: the most specific concepts which subsume it, and the most general concepts which it subsumes. Two independent searches are conducted to find these.

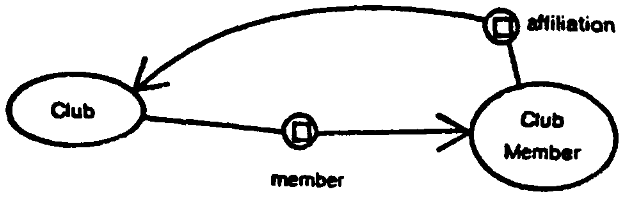


FIG. 5. A CYCLIC DESCRIPTION

5.1 Searching the Network

The search for subsumers starts at the top of the lattice and performs a depth first traversal, testing each node to see whether it subsumes the concept. If it does not, none of its children could either, so the subtree below it need not be searched. If it does, it is remembered as a subsumer and its children are searched. If none of its children subsume the concept, then it is the most specific subsumer (in this subtree) and a superconcept cable is established between the concept and its newly found subsumer.

While at first glance it might seem that this search need only include the subtree below a concept's initial classification, this is not the case. As shown in Figure 6, the concept AthenaCorporation is known to be a kind of Corporation, but not a kind of Woman'sOrganization. In order for the relationship to Woman'sOrganization to be found, the search must encompass more than just the subtree below Corporation. Note that if all possible conjunctions of concepts were represented in the network this would not be necessary, as the concept Woman'sScientificCorporation would exist as a subconcept of both ScientificCorporation and Woman'sOrganization. It is still not necessary to search (potentially) the entire network, as the relationship of the subparts of AthenaCorporation to the rest of the network provide some constraints on what must be searched, but this information is not currently used.

More constraints are available to limit the search for a concept's subsumees. First, because the subsumption relation is transitive, all subsumees of a concept are also subsumed by its subsumers, so it is only necessary to consider the subconcepts of the superconcepts just found. Second, since a concept's subsumees must have all of the roles that the concept has, it is only necessary to search those subtrees in the network consisting of concepts which have all of those roles. The appropriate

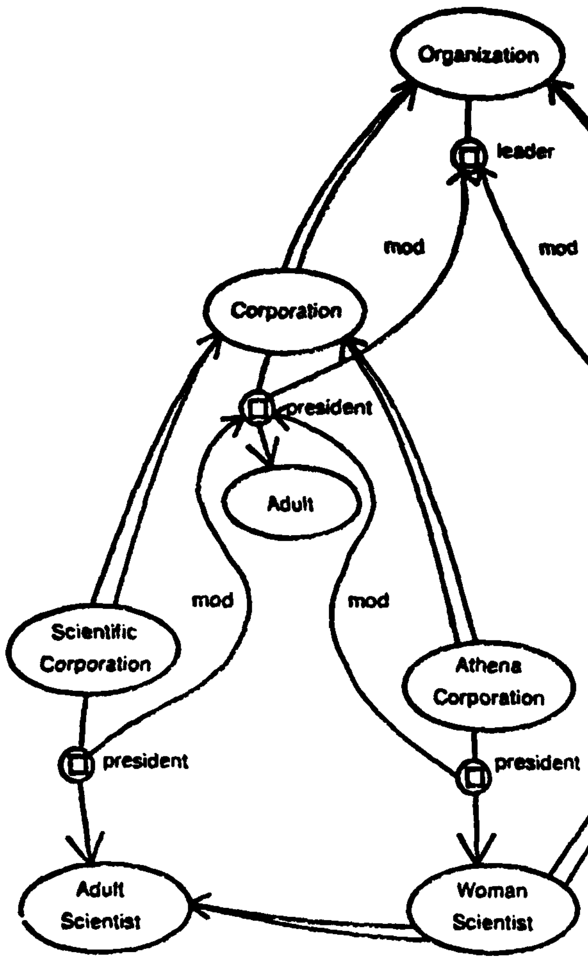


FIG. 6. SEARCHING THE NETWORK

subtrees are traversed, and the concept being classified is established as a superconcept of the highest concept in each subtree which it subsumes, if any.

5.2 Determining Subsumption

Both traversals of the network perform the same test at each node to determine whether one concept subsumes another. In one traversal the potential subsumer is a concept already in the taxonomy, and the potential subsumee is the concept being classified. In the other traversal the reverse is the case. For the remainder of this section, the potential subsumer will be referred to as Super, and the potential subsumee as Sub.

The first condition for subsumption is that Super is not a descendant of any natural kind concepts of which Sub is not a descendant. This test can be made without examining the subparts of either concept and is performed first. The remaining subsumption requirements express conditions on the relationships between corresponding subparts of the two concepts, so the next step is to determine and check these correspondences. Concepts have two different kinds of subparts: roles and role set relations. In order to determine the relationships between role set relations, the relationships between the roles involved must be known. Thus, the role subsumption relationships are established and checked first.

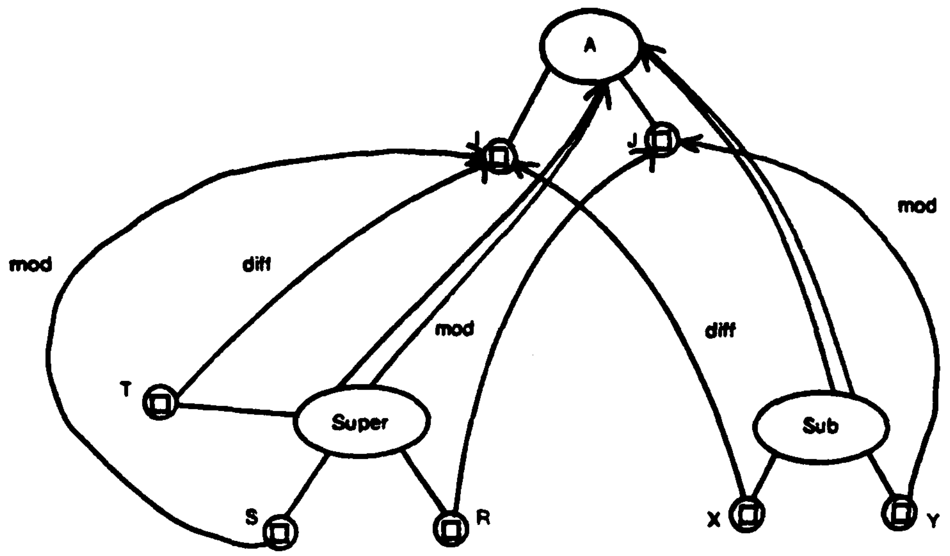


FIG. 7. EVIDENCE FOR ROLE CORRESPONDENCES

Given a role A of Super, and a role B of Sub, the fact that all of the facets of A subsume facets of B is not enough to determine that there is a relationship between the <u>meaning</u> of A to Super and the <u>meaning</u> of B to Sub. (E.g., just because a Person has a Mother which must be a Woman and a Man has a Wife which must be a Woman does not mean that the Mother role of Person is in any way related to the Wife role of Man.) The only way such similarity of meaning is represented in KL-ONE is in terms of the modification and differentiation relationships between roles in the network.[3] If a role of Super and a role of Sub are both descendants of a common role (which necessarily belongs to a concept which is a parent of both Super and Sub), then there is evidence for some relationship between the two roles. As shown in Figure 7, both the R role of Super and the Y role of Sub modify the J role of A, implying that R has the same meaning to Super as Y does to Sub so that, were Super to actually subsume Sub, role Y should modify role R. Similarly, the X role of Sub differentiates role I which is modified by the S role of Super, so role X should differentiate role S. Since there is no information available that role T of Super is the <u>same</u> subset of I as is X of Sub, no relationship can be inferred between X and T.[4]

Having discovered the relationships between the roles of Super and Sub, the role subsumption criteria can now be checked: for each role of Super, there must be one role of Sub which modifies it, and possibly others which differentiate it; if one role of Super differentiates another, there must be a similar differentiation relationship between the corresponding roles of Sub; the number restriction of each role of Super must specify a range which include the number restriction of the role of Sub that modifies it, as well as any roles of Sub which differentiate it; and the value description of each role of Super must subsume the value description of each of the roles of Sub which modify or differentiate the role.

The subsumption criteria for RSRs is that each constraint specified for Super must hold for Sub. The classifier currently demands that the constraints be explicitly stated on Sub, that is, each RSR of Super must

---

[3]Roles also have names, but these are only notational conveniences and carry no semantic information, except perhaps to processes external to the network.

[4]There are actually two possible interpretations of differentiation; the decomposition may be defined by the structure of the differentiating roles (as in the decomposition of a limb role into arm and leg roles having value descriptions of Arm and Leg, respectively, meaning that the arms are exactly those limbs that are Armish), or it may be unspecified (i.e., the value description expresses a feature true of the differentiated role but does not completely define the differentiation). In the first case the classifier would have enough information to match up the differentiating roles. However the two cases are not currently distinguished by KL-ONE, and the classifier assumes the second interpretation.

subsume some RSR of Sub. Unlike roles, the meaning of a role set relation to a concept is completely represented in its structure. Furthermore, the relationships allowed in KL-ONE between role set relations are not as rich as those between roles, and are expected to change in the near future. For these reasons, the correspondences between role set relations are determined solely on the basis of their structure. For each role set relation of Super a search is performed for a role set relation of Sub which it subsumes.

Two conditions must be met for a role set relation of Super (SuperRSR) to subsume one of Sub (SubRSR): the relation expressed by SuperRSR must subsume the relation expressed by SubRSR, and the parts of Super related by SuperRSR must subsume the parts of Sub related by SubRSR. These parts are expressed as chains of roles specifying a path from the concept to which the RSR belongs to a role within its description, which is the actual part. One rolechain subsumes another if the two chains are of equal length, and each role in the sub chain modifies the corresponding role in the super chain.

If the role set relations in question are role value maps, then if the relation type of SubRSR is EQUAL, then one of the two rolechains of SuperRSR must subsume one of the rolechains of SubRSR, and the other must subsume the other. If the relation type of SubRSR is SUBSET, then the relation type of SuperRSR must also be SUBSET, the left hand rolechain of SuperRSR must subsume the left hand rolechain of SubRSR, and similarly for the right hand rolechains.

If the role set relations are para-individual role set relations, then SuperRSR's para-individual must subsume SubRSR's para-individual. However, because para-individuals are parameterized and individual, it is not possible for there to be a subsumption relationship between them directly. While KL-ONE allows arbitrary specialization of a concept when it is para-individuated, the classifier assumes that the para-individual is merely a functional instantiation of its generic parent (i.e., it contains no additional structure), and so tests the subsumption relationship between the parents (as in Figure 3)[5] . Analogously to role value maps, the rolechain stemming from each coref role of SuperRSR's para-individual must subsume the rolechain stemming from the corresponding coref role of SubRSR's para-individual.

[5]The situation is actually somewhat more complex than this, as it is possible for the para-individual to inherit information from the concept to which it belongs (such as when a coref-role points to a role with a value description). If necessary, a new generic concept that contains this information is created and classified, and used to test subsumption.

5.3. Identical Concepts

If a concept which is identical to one already in the taxonomy is entered into the network and classified, then it will both subsume the existing concept and be subsumed by it. In order to correctly represent the relationship between these two concepts in the network, they must be merged.[6] The first time a cable is established between a concept being classified and an existing concept, the classifier tests whether the two concepts are identical[7]. If they are, all pointers to the new concept are changed to point to the existing concept, any rolechains going through roles of the new concept are rerouted through the corresponding roles of the existing concept, any nexus description wires or attached data are moved, and the new concept is discarded. This procedure can result in a loss of information if some process external to the network uses names to access parts of the structure. In order to help such external processes deal with this, the classifier provides as part of its result a list of concept substitutions it has made.

5.4. Interface to the Realizer

The Consul realizer (see Bill Mark's paper "Realization", page 78, and also [7]) is responsible for insuring that all incoming descriptions are attached to the nexuses they describe. While the classifier takes care of this for cases where a description describes a nexus by definition (i.e., some subconcept of the description already describes the nexus), external knowledge is required to determine whether a nexus described by some existing concept happens to also be described by a non subsuming incoming description. Making this determination in general is an intractable problem, so the realizer is only invoked in a restricted set of circumstances. Currently, the realizer is able to determine that two descriptions describe the same nexus when the only differences between the descriptions are coreferentiality constraints or structural descriptions. The classifier detects this occurrence when it is testing the subsumption relationship between two descriptions and invokes the realizer. The realizer uses external knowledge to decide whether the nexuses described by one description meet the structural constraints of the other, and performs the necessary attachments.

---

[6]Circularities in the superconcept hierarchy are not allowed in KL-ONE.

[7]_Identical_ here means structurally identical. Differences in names are ignored, as are attached data.

6. Conclusion

The classifier is currently being used by the Consul system as described in Section 2 of this paper. It is also used by the Consul explainer to determine why one structure does not subsume another, in the process of explaining why Consul was unable to redescribe a user request as an invocation of a service operation. It is expected that the classifier will be used by more systems in the near future, and that it will eventually be incorporated into KL-ONE.

Some work still needs to be performed on the implementation. There are some cases where ambiguous relationships between two concepts will cause a subsumption relationship to be missed. The algorithm must be modified to try all possible relationships in these cases before deciding that the subsumption relationship is not present. The algorithm must also be extended to properly handle cyclic structures as described in Section 5. Even when these shortcomings are rectified, the classifier will need attention periodically as KL-ONE is still evolving. Since the classifier is the embodiment of the KL-ONE semantics, it must evolve with the language.

The ideas and techniques presented here should be applicable to any knowledge representation system which supports analytical definition of terms. A classifier for systems which do not provide domain independant epistemological primitives would have to be domain specific, as it must contain knowledge of the meaning of the description building operations of the language. Such a classifier would still provide all of the advantages of the KL-ONE classifier, but in a restricted domain.

Acknowledgements

Many of the ideas presented in this paper were developed in discussions with members of the KL-ONE community, especially Bill Mark, Ron Brachman, and Rusty Bobrow.

REFERENCES

[1] Ronald Brachman (1978)
A Structural Paradigm for Representing Knowledge. Technical Report, Bolt Beranek and Newman Inc.

[2] W. A. Woods (1979)
Theoretical Studies in Natural Language Understanding. Annual Technical Report. BBN Report No. 4332, Bolt Beranek and Newman Inc.

[3] William Mark (1981a)
"Representation and Inference in the Consul System," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*.

[4] David Wilczynski (1981)
"Knowledge Acquisition in the Consul System," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*.

[5] David Israel and Ronald Brachman (1981)
"Distinctions and Conclusions: A Catalogue Raisonne," in *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*.

[6] Ronald Brachman (1980)
"I Lied About The Trees." Unpublished Manuscript.

[7] William Mark (1981)
Consul Note 12: "The Consul Realizer". (Also in BBN Report No. 4842, "Proceedings of the 1981 KL-ONE Workshop".)

A KL-ONE Based Knowledge Representation Kernel and Closing Remarks

Ron Brachman

Fairchild Laboratory for Artificial Intelligence Research
Palo Alto, California

(Presented during the main conference)

Now is as good a time as any to sit back and reflect on what has happened as KL-ONE has developed over the past few years. From this Workshop it is obvious that, whatever KL-ONE really is, it has succeeded at one important scientific task: it has drawn together a large number of very capable people to discuss issues of critical importance to knowledge representation and Artificial Intelligence in general.

Its service as a focal point for energetic pursuit of the important issues of the day may be the most significant contribution of KL-ONE - it is certainly the most easily articulated. Technical details are fleeting, and as we all know, it is often hard even to enumerate enough of those technical details to pin down what it is that KL-ONE is. Several of us (including Danny Bobrow, Richard Fikes, Austin Henderson, Hector Levesque, and Mark Stefik) have been musing over this point recently. With a surprising amount of hard work (considering that we thought that we knew what KL-ONE was), some fierce head-scratching, and a lot of mind-changing, we have begun to understand what the kernel of KL-ONE is all about.

The technical details of our rational reconstruction are covered in large part in the report on the technical discussion on Assertions (Section 1.1, page 8, in this Proceedings). I will take the opportunity here to touch on some of the high points of the KL-ONE kernel, to some extent indicating the direction that I think research on KL-ONE should be going in the future.

Some observations

Our investigation into the foundations of representation began with an observation of some inconsistencies in our story about KL-ONE. On the one hand, it was easy to consider the number of subConcepts below a Concept as literally representing the number of subtypes of the supertype. For example, if we found INDIAN-ELEPHANT and AFRICAN-ELEPHANT below ELEPHANT, a rational explanation might be that this represented the fact that there were exactly two types of elephant (some story needs to be told here about whether this is with respect to one world, all possible worlds, etc.). On the other hand, we have tried to emphasize KL-ONE's support of compositional Concepts (see [Israel and Brachman 81]). One consequence of handling structured Concepts in the way we have advocated is that any Concept is considered to have an infinite number of subConcepts below it - all complex terms formable with that Concept as head. It was clear that our networks were being asked to serve different purposes at the same time.

At the same time we discovered that there were some issues arising out of our network notations that didn't really make sense. In our networks we had the notion of a "local Role" - you could fetch all of the Roles of a Concept that were attached to it directly, rather than by inheritance from some more general Concept. With a few moments' thought, one can see that a differentiation between local Roles and those that are inherited is an artifact of the node-and-link-style language that we have used to express KL-ONE Concepts. "Mods" links were an attempt to reflect in structure the fact that a Concept descended from another was supposed to have the very same Roles as that superConcept. We couldn't easily have a physical structure in two places at once, so in the descendant's case, we created a pointer to the Role that was considered inherited. Once we allowed local modifications to that Role, we began to visualize the pointer-plus-modifications as a full-fledged Role. The net result was that we began to talk about an *implementation* of a certain conception as if it were the conception itself.

Focus on functionality

Given the above sorts of confusions, we chose to address the problem in the large. We asked ourselves these questions: 1) what multiple purposes were we asking our network language to serve?, and 2) which of our concerns were implementation concerns, and which were about the functionality of a representation language, independent of its implementation?

We have only begun to sort out the major types of functionality we want for our representation system. Roughly speaking, we have two principal components - one that deals with matters of terminology, and one that handles assertional concerns. Our experience with KL-ONE has taught us that an effective way to segment the knowledge representation task is to deal with strictly terminological knowledge on its own grounds. For one thing, keeping strict matters of terminology to themselves admits the idea of a classifier, which maintains analytic subsumption relationships (i.e., which other descriptions a description subsumes by virtue of meaning). Also, conceptual composition - the ability to form new compound terms from an existing terminological basis - has emerged directly from our consideration of purely definitional knowledge distinct from asserted facts about the world.

Interestingly enough, one could imagine implementing both a terminological competence and an assertional one in a "semantic network" type framework. Both kinds of information might easily benefit from "inheritance" and transitive inclusion relationships (in the one case, a definitional inclusion relationship - "subsumption" - and in the other, a set inclusion one). A brief look at some of the confusions in semantic net history tells you how seductive this similarity is: the infamous "isa" link has wandered back and forth from meaning simple set inclusion to conceptual composition to even a "general-purpose inheritance link", with which one can specify arbitrary patterns of features to be inherited.

In KL-ONE, we have traditionally used a network-style notation to represent our classification hierarchy and the internal structure of Concepts. The network language - "Structured Inheritance Networks" (SI-Nets) - emphasizes some features of KL-ONE Concepts and helps us visualize the taxonomic relations among Concepts. However, we should remember that SI-Nets have properties of their own that are not relevant to KL-ONE; SI-Nets are an implementation medium for a knowledge representation language like KL-ONE, but not KL-ONE itself. We should always be on the lookout for SI-Net issues in disguise, lest "Mods" links and "local Roles" become diversionary problems. It must also be said that there is a definite place for concerns at the SI-Net level. As we mentioned above, it is quite possible that the notion of say, a "subConcept" can be used in support of both terminological and assertional reasoning. We should just be careful to acknowledge which level we are addressing.

Other aspects of the framework

A few of the details of our reconstruction of KL-ONE should probably be summarized before closing. One central point is that analytic description subsumption is the primary consideration when dealing with the terminological component of a representation system. However a terminological machine ("Tbox" as we have called it elsewhere) is constructed, its job is to answer the question, "does description d1 subsume description d2?" The taxonomic lattice that we have constructed out of SI-Nets is one way to implement this capability, but not the only one. Notice also that inheritance is strictly an implementational consideration with respect to subsumption. It is exactly because one description subsumes another that the latter seems to "inherit" properties from the former. Since we are used to dealing with labelled nodes in our networks - i.e., we communicate with names rather than with full descriptions - it seems as if we are learning something when we find that ISOSCELES-TRIANGLE "inherits" three-sidedness from TRIANGLE. But, in reality, ISOSCELES-TRIANGLE is just a label for a more complex description, which when seen in its entirety (polygon with three sides, the lengths of two of which are equal), makes it obvious that we are talking about a three-sided figure.

The lesson here is that the right way to approach the Tbox is not to think of "adding" a concept, or putting something in a place in the network, but to think of constructing a description out of a fixed set of compositional description-forming operators, such that the description itself is a "place" in the subsumption lattice. Simply mentioning a description gets you to a place in the terminological space such that all of the "right" inferences (subsumption ones, in particular) follow from being "at" that place.

The kinds of descriptions (KL-ONE Concepts) we have in mind for the Tbox are roughly akin to noun phrases in a natural language. Sentences are the province of the Assertional Component (Abox); the "terms" used in the sentences come from the Tbox. Our Concepts are more like predicates than like sentences, and thus correspond to noun phrase rather than sentences (this is not to say that the Concepts are predicates - our feeling is that the Tbox deals with Concepts, the Abox with predicates, and there are mappings from one to the other).

The Tbox is actually a set of rules for forming Concepts out of other Concepts and a set of rules for determining subsumption relations among Concepts. We have been looking back at KL-ONE to try to determine exactly what the implicit rules there were. For the most part, these are obvious - we have Concept-formation through multiple-superConcepts, for instance, with the accompanying statement that if x is describable by A and is describable by B, then it is describable by the compound Concept, A&B. Other ways of forming Concepts include "tightening" a Value Restriction on a Role, tightening a Number Restriction, QUA, adding a new Role to a Concept (this is actually a special kind of Differentiation), and deriving Individual Concepts and Primitive Concepts (formerly called "magic"). Subsumption can be derived just from the internal structure of Concepts in all cases where they are formed compositionally, but not in the case of Primitive introduction. The latter is necessary to get started, and to form terms corresponding to those with no necessary and sufficient conditions. *THING, the most general Concept,* is primitive. We have also determined that Role Differentiation comes in both compositional and non-compositional varieties, with the former being the way to get Roles like "male children" directly from the Role "child" and the V/R "male", and the latter being the way to introduce new Roles without necessary and sufficient conditions. Work on the logic of the Tbox is continuing, and we hope to report on results sometime soon.

Conclusion

It is obvious from the attendance at the Workshop, the diversity of points of view expressed in the position papers, and ongoing research (like that reported above) that work on KL-ONE is continuing vigorously, both on a variety of issues, and with a wide spectrum of approaches. It is nice to see such a diverse community at work on common goals, and I would like to encourage non-experts to (continue to) participate in the research community, and experts to try harder in communicating with non-native KL-ONE speakers (there are so many more of them than there are of us!). It is also good to see alternative ways of thinking about the language beyond our network-style pictures - one lesson we have learned in our reconstruction is that it is easy to become carried away with issues of syntax and implementation.

Also, please use the mailing list at the back of this Proceedings. The more we communicate, the better off we'll be.

4. POSITION PAPERS

Prior to the Workshop, we asked each potential attendee to submit a position paper describing his/her interest in KL-ONE. In general, each group of participants responded jointly (as we suggested). However, some individuals submitted their own position papers. This chapter contains the papers we received. The actual text of our request follows:

> Write a 1-3 page paper on your group's opinion regarding the following issues:
>
> If you use KL-ONE, either on a computer or on paper, tell us what gives you the most trouble with it: what is hard to represent? why? what features don't you like? Also, tell us about what you find appropriate in KL-ONE for representing your domain, and what features of the language you think are best.
>
> If you are considering using KL-ONE for some reason, tell us (in some detail) why KL-ONE might be appropriate to the problem area to which you would like to apply it. Have you considered other representation languages? If so, what can you tell us about how they compare with KL-ONE?
>
> If you have a general interest in representation, or are a "critic", tell us about what you think are the most significant concerns in representation work, and what you know about how well KL-ONE and other languages address these concerns.

Robert J. Bobrow (author), Candace Sidner,
David Israel, Jim Schmolze, Bill Woods,
Brad Goodman, Madeleine Bates

Knowledge Representation for Natural Language Understanding,
Bolt Beranek and Newman Inc.

THE NEED FOR ROLE SET RELATIONS IN KL-ONE

In addition to the use of KL-ONE to represent the syntactic and semantic structures needed to interpret the parses produced by the RUS parser[1], I have also been interested in the strengthening of the RSR (SD?) component of KL-ONE. This stems, in part, from work done by Candy Sidner and myself on the representation of plans in KL-ONE[2]. In particular, we have found that the representation of plans would be facilitated by the ability to represent orderings and other relations among the Roles of a Concept.

Thus, we argue for the need to maintain and expand the RSR facility in KL-ONE. In particular, we would like to see the n-ary relations which are part of RSR's (except for the RVM type of RSR) become part of the "structural component" of a concept, as well as part of the "assertional component". We will explain this distinction below.

Central to the view we are presenting is a distinction to bear in mind when one considers the elements of KL-ONE, the assertional and the structural aspects of KL-ONE. These relate to the difference between the use of KL-ONE to assert both universal and contingent properties of objects, relations, etc., and its use to "name" and thereby provide access to the conceptually relevant structural components of objects, etc.

---

[1]See Bobrow&Webber "Knowledge Representation for Syntactic/Semantic Processing" in the first AAAI proceedings.

[2]See the position paper by Sidner and Bobrow in this collection (page 182).

The original version of RSR's as (partially) designed and implemented included the notion of a distinguishable component of a concept which represented an n-ary relation among the roles of that concept. This was provided in part to give a basis for the quantification needed to define the way in which PI's are associated with a concept.

It was also provided in order to allow the description of structures which have an ordering relation among their parts. Thus, it would be possible to describe a LIST as having one Roleset MEMB, and having a linear ordering relation NEXT whose domain was that Roleset. This allows the elements of the list to be the fillers of the MEMB roles, and to have properties asserted of them independent of their position (or number of occurrences) in the list. Thus, in a list which represents a queue for some process, the MEMBs are all processes which will (should) eventually be run. The relation NEXT provides a means of accessing list elements in the order determined by the list itself. Note that this is a property of the Role and not of the filler, since a given item may appear more than once on a list, and the NEXT item depends on the position (Role) within the list.

While the LIST example does not bring out the use of the relation NEXT as part of the assertional mechanism, consider a case where a company is described as having a Roleset OFFICER and a Roleset DEPARTMENT, and there is an assertion that for every DEPARTMENT there is an OFFICER such that some relation (expressed by a ParaIndividual, such as REPORTS-TO) holds between the DEPARTMENT and the OFFICER. While this seems to be a simple case of AE (that is, "for all ... there exists ...") quantification applied to Rolesets, it would be valuable to give a name (say RESPONSIBLE-OFFICER) to the relation (the Skolem function in this case) implied by the AE quantification. If such named relations were accessible just like Rolesets, it would be possible to ask for the RESPONSIBLE-OFFICER for a particular DEPARTMENT, or the set of DEPARTMENTs for which a given OFFICER is the RESPONSIBLE-OFFICER. Note again, that this is a relation between the Rolesets and not the fillers. If Jones is both the Financial Vice President and the Purchasing Officer, he might have different departments reporting to him in each Role.

One common view of RSR's is that they simply provide (intensional) assertions about the components of an entity described by the concept containing the RSR. In this way they are distinguishable from Rolesets, whose primary purpose is to provide a means of imposing a structured view of an entity, providing "names" or intensional means of describing components and entities which are related to the central entity described by the concept as a whole.

This distinction is rather problematic, however, when one takes into account the number facet on Rolesets. That facet appears to make an assertion that any entity of the given type will have some number (constrained by the number facet) of components of the type represented by the Roleset. In a similar but reverse direction, the features of RSR's which are normally viewed as providing for "quantification" of assertions are actually valuable for providing a means of referring to components of the entity described by the concept.

The notion of n-ary relations as part of RSR's has been neglected, in part because of the fact that until recently there has been no critical need for representing structures which have non-trivial internal orderings. It has also been avoided because there was no concrete proposal to provide a descriptive characterization of the relations, e.g. whether they were partial orderings, functions from one Roleset onto another, etc.

In our work on the representation of plans in KL-ONE, we have come up with clear cases of structures which seem best represented with internal orderings, and in fact with many orderings that interact. We have some ideas for declarative characterization of relations, and have also uncovered a need for a structuring of RSR's that corresponds to the notion of Diffs in Rolesets -- in fact, combining with Mike Freeman's note on strengthening of theories, it suggests that RSR's be treated in a fashion very similar to Rolesets.

Ron Brachman
Fairchild Research and Development

THOUGHTS ON KL-ONE

Having been as close to KL-ONE as I have, my general impression of it is, naturally, positive. In particular, I am most proud of the way that the KL-ONE community has developed and of the research methodology that has been the foundation of work on the language. I think that the people who have contributed to the language/system development over the last four years have taken as broad and intellectually honest an approach to knowledge representation as anyone else in the field (and this is despite the fact that the work initially emphasized natural language so heavily).

We have had our shortcomings, however, in particular in the lack of communication with the world outside of our small (and usually isolated) community. If it were not for the strong support of ARPA, and the vigorous activity of the community actively using and discussing KL-ONE, I think it would have fallen on hard times some time ago for lack of telling anyone about ongoing research and any claims we had to offer. I don't know if we have been worse than most other research communities, but we have time and again failed to write down our conclusions, and have thus re-walked the same research paths in quite a few cases (I have certainly been as guilty as anyone on this score). I think that we as a community have a tremendous amount to offer the rest of the knowledge representation world, and really should try harder to share our work with them. The proceedings of this Workshop should be a good start.

Now, with respect to more technical issues, I think that we have developed a language that is cleaner than most, and as communicatively powerful as any (despite its perhaps inferior expressive power - at least with respect to predicate logic - at the moment). The strongest points of KL-ONE are its acknowledgement of the need for definitional as well as contingent representations, the whole idea of an analytic classification hierarchy with an implicit classifier, the notion of RoleSets and the subsumption hierarchy for Roles (with both Role restriction and differentiation), and at least the idea of SD's (despite the lack of technical detail on their structure) as supporting the meaning of functional roles within Concepts.

On the other hand, there is an incredible amount of work left to do; we have just waved our hands at the representation of contingent information, let alone the default/prototype kind that is so important in reasoning, and SD's still remain a mystery. There are still open questions as to what power is required of the classifier (for example, how much, if any, number theory do we need to include in order to classify with Number Restrictions that are more powerful than the ones we have), what the relation between Concepts and Roles is, and how to handle multiple worlds (we have failed to say anything about including worlds inside of descriptions, for example). I would love to see more work on the JARGON idea - there is something important in talking to a representation system in terms of the domain to be represented rather than in terms of "RoleValueMaps" and such (I thank Hector Levesque for making me realize the importance of this).

All in all, though, I think that we have been headed in the right direction. Some of us in our own local KL-ONE-klatch have been looking back and trying to understand the differences between the node and link structures we are used to thinking in and what we REALLY wanted to say (this is reported elsewhere in this proceedings) - I am constantly impressed with the way we have torn apart old ideas and found that they were, in the end, very close to "right", although not usually for any explicitly understood reason. I sure hope this continues.

Jeff Conklin and David McDonald

Department of Computer and Information Science,
University of Massachusetts at Amherst

SPATIAL RELATIONSHIPS IN KL-ONE

Our goal is to generate English descriptions of scenes of suburban houses, these being the principle subjects of the local research on vision. Our system takes as input a non-linguistic 3-D representation of a particular scene and generates a natural-sounding paragraph description from it.

Our concern with KL-ONE is in representing visual information about objects and their relationships. There are three main kinds of relationships we need to capture. Some relationships are tightly bound to some object: structural ("part-of") relations, e.g. the Roof is Part-of the House; and "schematic" spatial relations, e.g. Fences Surround Houses. By "schematic" we mean that a fact is regular and predictable enough to be encoded as part of a generic concept.

Other relationships (mostly spatial) are not part of any scene concept: e.g. the Person is In-Front-Of the House.

We represent a structural relation as a role node on the concept node of the major object (Figure 1).

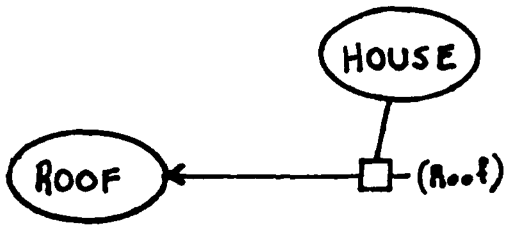


FIG. 1.

Schematic spatial relations are concept nodes in the Structural Description of the generic concept, and specify how the roles relate (Figure 2).

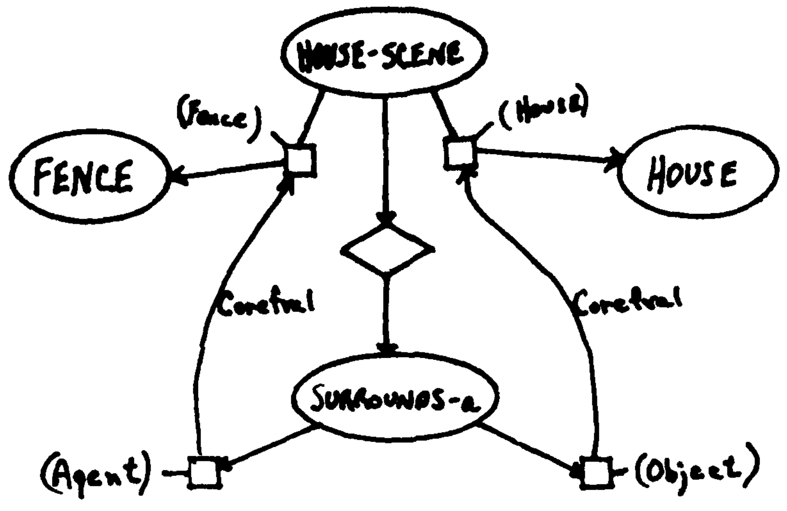


FIG. 2.

Any non-schematic spatial relations are individuated under some more abstract generic concept, e.g., in the net in Figure 3 the generic In-Front-Of is restricted only to having "Things" as its Agent and Object role fillers. This is not to say that In-Front-Of is in itself a non-schematic relationship: a schematic case, e.g., Mailboxes are <u>In-Front-Of</u> Houses, would be handled as in Figure 2.

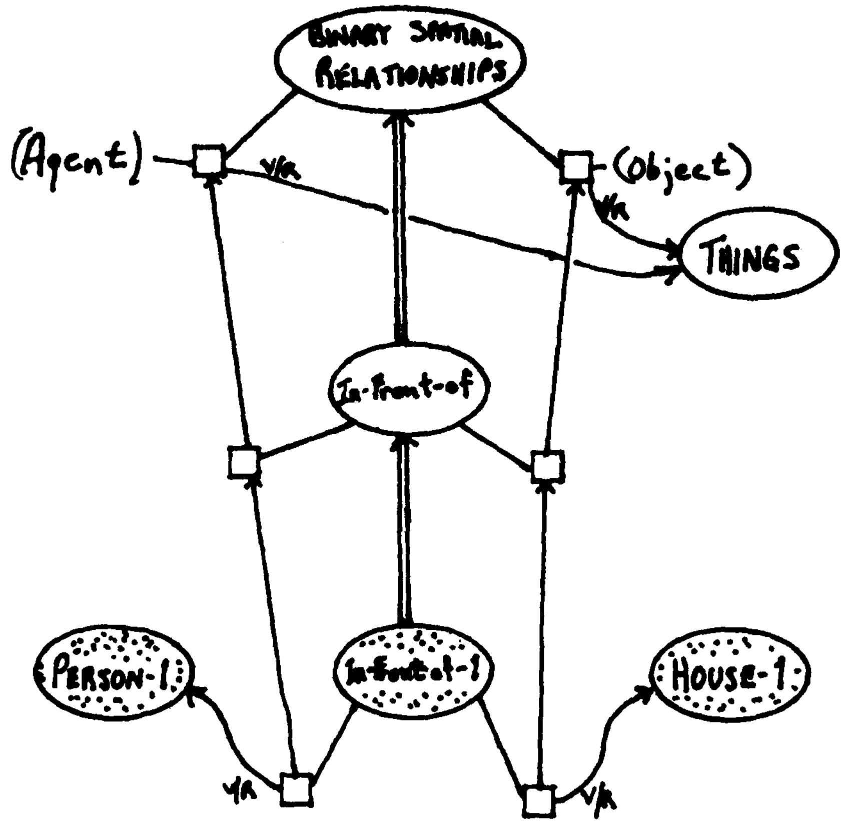

BINARY SPATIAL RELATIONSHIPS
(Agent)
V/R
(Object)
V/R
THINGS
In-Front-of
PERSON-1
In-Front-of-1
HOUSE-1
V/R
V/R

FIG. 3.

Tim Finin, Richard Duncan, Hassan Ait-Kaci

Franz KL-ONE Translation Project,
University of Pennsylvania

We are involved in bringing up a version of KL-ONE in FranzLisp for use on a VAX computer. Our approach has been to build a general purpose inter-dialect Lisp translation system with a well-developed Interlisp-to-FranzLisp component. Thus, we hope to keep up with new versions of Interlisp KL-ONE[3].

[3]For a more detailed discussion of this project, please refer to the paper presented by Tim Finin, entitled "Translating KL-ONE from Interlisp to FranzLisp".

Alan Frisch, James Allen

Natural Language Project,
Computer Science Department,
University of Rochester

Last year at the KL-ONE Workshop, we presented a position paper. The paper raised some representation issues that are often neglected but seemed crucial based on our work in natural language processing. The issues addressed at that time were:

1. the indexing of concepts by contexts, i.e. by beliefs of agents, by time, and by situations;

2. the explicit representation and inference about coreference of concepts; and

3. the precise specification of retrieval mechanisms and inexact matching of concepts.

These issues have occupied our attention throughout the last year. Papers have been written that deal with time [2, 3], coreference [4, 5], and the specification of retrieval mechanisms [5, 6].

The natural language project now consists of 8 researchers (Gary Cottrell, Hans Koomen, Diane Litman, Lokendra Shastri, Steven Small, Marc Vilain, and ourselves) actively implementing a system to partake in English dialogs [1]. The system is organized as a set of communicating concurrent processes, one of which is a knowledge base.

The processes communicate in a representation language whose notation is a syntactic variant of first-order predicate calculus. As in KL-ONE, it uses a fixed set of epistemological predicates that are defined by a fixed set of domain-independent axioms.

The knowledge base stores representation language sentences and provides retrieval facilities for the other modules to use in the course of performing their task-specific inferences. We view knowledge retrieval as a limited form of inference operating on the stored knowledge. The knowledge base only performs those inferences for which it has adequate control knowledge to perform efficiently.

We have developed a methodology for using a meta-language to specify limited inference in the representation language. This technique has been used to specify a retriever that cannot perform arbitrary logical inference but can perform inferences typically found in semantic networks such as inheritance [5, 6]. The specified retriever has been implemented and is currently being used by the dialog system.

Some aspects of our work has been influenced by KL-ONE. Clearly, the use of a fixed set of epistemological predicates is consistent with the KL-ONE philosophy. Our representation has also been influenced by KL-ONE in that there are predicates used to build concepts and predicates used to make assertions about these concepts. On the other hand, the representation is similar to Hendrix's partitioned semantic networks in that it treats all concepts as individuals at the bottom level of a type hierarchy. In fact, the subset of our representation language that deals with describing concepts can be viewed as an axiomization of the Hendrix system.

Other aspects of our work diverge significantly from KL-ONE. Our use of the notation of first-order predicate calculus is a crucial methodological commitment that removes all problems of notation design. On the other hand, it forces no representational commitments. Our primary criticism of KL-ONE is its lack of a well-defined inference system and a retriever based on this inference system. (This deficiency may be due to the fact that the assertion language is still in its infancy.)

REFERENCES

[1] J. F. Allen (1980)
"The Rochester Natural Language Understanding Project." 1980-81 Computer Science and Computer Engineering Review. Computer Science Dept., Univ. of Rochester.

[2] J. F. Allen (1981)
"What's Necessary to Hide?: Modelling Action Verbs." Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics, July.

[3] J. F. Allen (1981)
"Maintaining Knowledge about Temporal Intervals." Proceedings of the Seventh International Joint Conference on Artificial Intelligence, August.

[4] J. F. Allen and A. M. Frisch(To appear, 1982)
"What's in a Semantic Network Matcher?"

[5] A. M. Frisch (1981)
"A Formal Study of Knowledge Representation and Retrieval." Ph.D. thesis proposal, Computer Science Dept., Univ. of Rochester, June.

[6] A. M. Frisch and J. F. Allen (To appear, 1982)
"Knowledge Retrieval as Limited Inference."

Austin Henderson, Richard Fikes, Mike Williams,
Fred Tou, Ben Cohen

Cognitive and Instructional Sciences Group,
and Daniel Bobrow, VLSI Group,
Xerox Palo Alto Research Center

Richard Fikes

IMPLEMENTER: Richard has created a version of KL-ONE in Smalltalk. This "KloneTalk" system has a display interface which has been tuned to permit easy interaction with networks.

INTERESTED PARTY: Richard is active in a group which is working on a redesign of KL-ONE.

USER: Richard uses KL-ONE for modelling office procedures.

Austin Henderson

INTERESTED PARTY: Austin is active in a group which is working on a redesign of KL-ONE.

PROSPECTIVE USER: Austin intends to use KL-ONE to represent objects in an interface design environment.

Mike Williams and Fred Tou

USERS: Mike and Fred use KloneTalk to support studies of Reformulative Information Retrieval. The epistemology of KL-ONE is central in this effort.

Ben Cohen

INTERESTED PARTY: Ben works on representing knowledge about typicality.

Danny Bobrow

INTERESTED PARTY: Danny is active in a group which is working on a redesign of KL-ONE.

POTENTIAL USER: Danny may use KL-ONE (or some derivative) to represent knowledge concerning the design of VLSI chips.

Hector Levesque
University of Toronto

The KL-ONE language, as it now stands, can be divided into two regions. Region A contains entities like Concepts, Roles and Structural Descriptions. Region B contains Contexts, Nexuses and Description Wires. My feeling on KL-ONE that it is is this division that makes the language interesting but that the distinction between the two regions has been blurred somewhat in practice. In what follows, I will elaborate on what I see as the nature of the two regions.

Region A, as I see it, should be a terminological space. By this I mean that the purpose of A should be to structure and organize the various components in terms of which knowledge will be represented. The key point here is that Region A itself does not express this knowledge. There is no sense in which the components of Region A can be right or wrong; they are merely terminological conventions. For this reason, the whole idea of removing or modifying the components of A is, at best, operating on A at the wrong level.

Region B, on the other hand, is the part of KL-ONE that is used to represent knowledge using the terms of region A. The idea is that one should be able to form, in Region B, a partial picture of a world, that can be refined as knowledge about that world is accumulated. A major requirement here is that it should be possible to represent knowledge that is very incomplete, much more so than what is allowed by the calculus of Nexuses and Description Wires.

So, to summarize, Region B is used to represent what is known (about a world) using a language defined in Region A. My feeling is that the definitional mechanisms of Region A should be kept completely separate from the assertional mechanisms of Region B since the purpose and standards of adequacy of each region are so different.

Bill Mark, Bill Swartout, Tom Lipkis,
David Wilczynski, Bob Lingard

CONSUL project,
University of Southern California,
Information Sciences Institute

The Consul system is intended to be a cooperative environment for users of online services (message sending, text formatting, etc.)[1]. As such, the system must interpret natural language requests, infer the service execution necessary to satisfy requests, provide help when asked and explanations when requests cannot be fulfilled, and acquire service-dependent knowledge interactively with the service builder. Our decision has been to use KL-ONE to represent all of the knowledge necessary for these processes in a single system knowledge base--the program's sole repository of updatable, examinable knowledge.

We tend to be "strict constructionist" users of KL-ONE, representing descriptions of objects as definitional KL-ONE structures classified in the system knowledge base. More precisely, we take descriptions to be "definitional" internal to the system (allowing us to use our classifier to put any incoming description in its proper place in the network), while realizing that most descriptions cannot in fact be definitional in a broader sense (i.e., from the perspective of an observer outside the system). Descriptions are attached to nexuses--"old style" ones representing objects in the world--via "realization", a process that combines the use of existing network relationships with procedures that may take into account extra-network knowledge in order to determine whether an incoming description describes objects (currently described in other ways) in the system. All objects exist in contexts: one or more of three belief contexts (representing beliefs of the system, user, and service builder), and one or more time contexts (representing "the course of events").

These KL-ONE representations are fundamental to the workings of all of Consul's basic processes: in fact, the processes have been formulated in terms

[1]This paper was presented during the technical discussions.

of the KL-ONE representation philosophy (e.g., the system's inferential process is a combination of redescription/reclassification and RSR-based realization; acquisition is the creation of new instantiations and relevant mapping descriptions; incorporating belief is realization in different belief contexts). We have found KL-ONE to be extremely useful as a guideline for the design of these processes. The major benefit of KL-ONE per se has been the strict compartmentalization of different aspects of world knowledge as different representational structures. For example, the idea of RSR's and ParaIndividuals as distinct from "normal" concepts has been key to the realization process, and thus to inference and belief incorporation.

The major difficulties we have had with KL-ONE have been the lack of a framework for quantification and the considerable ambiguity over the status of "individuals". Also, we have not been particularly successful in the representation of such "process" knowledge as the effect of an action, "potential" versus actual occurrences, etc.--but it's hard to call this a fault of KL-ONE. No one really seems to be able to represent this kind of knowledge satisfactorily and it seems unfair to ask a representation system to solve the problem.

Our overall feeling about KL-ONE, then, is that it's the best thing going (we did consider others), and that it currently needs extension rather than major change.

Eric Mays, Aravind Joshi

Dept. of Computer and Information Science,
Moore School of Electrical Engineering/D2,
University of Pennsylvania

Our interest in KL-ONE is generally due to a need to represent knowledge required for certain aspects of natural language interaction with dynamic data bases that is not currently available in the db system. Specifically, we are striving for a description of the data base not only in terms of entities, relationships, and attributes -- but also the process of change (i.e. how the db is updated), indexing events and assertions by time, maintaining history, and incorporating belief structures.

Although we believe that these requirements might be achieved by some suitable logic formalism, KL-ONE achieves a more natural conceptualization for the purposes of natural language interaction. The problem of indexing propositions by time seems to be intimately related to that of indexing by belief. We would hope to represent time and processes as concepts in the KL-ONE framework, using structural description for precedence.

A substantial question is whether a mapping could be made from KL-ONE to a standard dbms. It will most likely be the case that the coverage of the domain in KL-ONE is broader than that available in the db. In the long term, this would indicate areas for increased coverage by the dbms, but immediately raises problems regarding communication of coverage failures to the user.

Peter Patel-Schneider, Bryan Kramer,
Yves Lesperance, John Mylopoulos,
Hector Levesque, Andrew Gullen

AI group - PSN,
University of Toronto

Among the many important concerns in representation work, there are five that are of particular interest to us. They are:

1. the need for semantics for representation languages,

2. the commitment of a representation language to its own rules,

3. the adequacy of a representation language in representing concepts and interactions in many problem domains,

4. the need for new or different organizational primitives in semantic network type representation languages, and

5. the ability of representation languages to represent procedural knowledge.

Many representation languages are not concerned with giving their constructs a real meaning. In these languages a construct's meaning is determined by its use and nothing else. To us, such a representation language is really just a package of functions that implement a complex data base system. A true language for knowledge representation must have some commitment as to what particular constructs in the language mean. For example, KL-ONE is based upon a set of epistemological primitives each of which has an informal semantics. In this way all constructs in KL-ONE have an implied semantics and thus actually do mean something. In PSN, we have a set of constructs that have the same epistemological force as the KL-ONE primitives. However, these constructs are not primitive and we have assumed that their construction was correct. When we started investigating the formal semantics of our language we found that these constructs had properties that we did not expect. In order to determine the formal meaning of these constructs, we are trying to formalize the semantics of PSN and we believe that a formal semantics is desirable for all representation languages, both to give meaning to all constructs in the language and to show that there are no (pleasant or unpleasant) surprises in the language.

Another important concern is the commitment of a representation language to strong rules. Some representation languages do no have any rules but instead are based upon the idea of defaults. In these representation languages organizational knowledge for domains is there to be used only if nothing is known about a particular object in the domain or no other knowledge conflicts with it. Other representation languages, such as KL-ONE and PSN, have strong rules and all knowledge in them is tightly constrained by these rules. The problems faced by these two types of representation languages are quite different in that the unconstrained types have a very hard time actually representing rules in problem domains and the constrained types have to find ways of handling exceptions to their rules.

All representation languages claim to be able to represent a large class of problem domains. Each has a particular set of primitives that can be used to model certain types of knowledge. In our opinion, no representation language contains a complete set of primitives in that they can represent everything and there is no reasonable way of doing this. The best that can be hoped for in a static system is that its set of primitives form a closely integrated, conceptually clean group. One such set of primitives is the epistemological primitives of KL-ONE. Our approach to circumvent this problem is to design a representation language in which new primitives (at some level) can be defined, given a meaning, and added to the language. In this way, our representation language can expand, and also contract, as need be.

One particular type of primitive are the organizational primitives of semantic network type representation languages. We have been thinking about whether new or different primitives are needed to go along with class membership, class inclusion, attributes and their inheritance, and contexts. As far as we can see, these are currently considered adequate by many representation groups. We have not come up with any radically new organizational primitives but, in line with having an extensible representation language in PSN, have designed a class of organizational primitives having to do with relationships between classes which contain class inclusion and other related concepts such as set inclusion and similarity links as members.

As [sic] representation languages contain a great deal of machinery for representing declarative hierarchy. However, few contain very much about representing procedural knowledge. Most that do contain procedures use them as black boxes that cannot be examined or explained. We have always used procedures in PSN as a basic part of the language and have made them full-fledged objects that can be investigated and placed in organizational hierarchies just as declarative objects can. This means that we can have procedural knowledge in a problem domain and are comfortable in using it, a desirable property for any representation language.

We believe that these five concerns are very important in current representation work.

Rachel Reichman
Institute for Cognitive Science,
University of California, San Diego

I have recently joined the computer science faculty at the University of California, San Diego, and the university's Institute for Cognitive Science. Over the past few years, at BBN, I have designed a computational model for extended discourse engagement. The model is not yet implemented and is written in a predicate-calculus-like language with predicates like "EXPRESS," "INFER," and "INSTANCE," considered primitives of the system. In a fully implemented system these primitives would serve as function calls to a generator, semantic component, an inferencer, and/or other pragmatic modules in the system.

My goal at UCSD is to develop an AI center with natural language processing a major area of research. First steps towards this end have already begun with the university approving my purchase of an INTERLISP personal machine - I plan to get a Jericho. Having the Jericho, I will begin implementation of a computer conversational system by using my discourse module, KL- ONE, and other natural language components already designed at BBN. In the discussion below, I briefly overview the main features of this discourse module and those features of KL-ONE with which I will be most concerned.

The Discourse Model

The major feature of the module is to partition an ongoing dialogue into a set of distinct, but related and linked, context spaces. Context spaces are functionally defined and they contain information both explicit and implicit in the dialogue. All utterances within a given context space fulfill a single communicative goal of a speaker(s). In the work, a communicative goal is characterized as some thematic, functional relation between parts of a discourse.

In the context space theory, a conversation is viewed as a sequence of conversational moves wherein each move corresponds to a possible speaker communicative goal. A shift in communicative goal is reflected in the work by a shift in context space. "Challenge," "Interrupt," and "Support," are some of the characterized goals.

At any point of discussion only some context spaces are in the foreground of discussion while others are rendered a background role. A major task of the discourse module is to identify and continually track which spaces are currently in the foreground of discussion. The module performs this task by correlating a set of effects with all conversational moves. Major effects of a move are to (1) set up discourse expectations of moves most likely to follow; and (2) change the foreground- background role of preceding context spaces. All conversational moves also have a set of preconditions which specify the requisite discourse environment for the move to be well-formed at a given point of discussion, and they all have various modes of fulfillment. In the main, modes of fulfillment are characterized by abstract logical/semantic relations - for example, one way of supporting a proposition, P, is to assert a proposition Q, where (INSTANCE Q P) is True.

As a conversation progresses, the discourse module constructs a "model of the discourse structure," which includes its context space partitioning of the discourse, the current topic of discussion, discourse entities currently in focus, and a list of outstanding discourse expectations of moves to come. Currently, the module constructs a single model, not distinguishing between its own conception of the discourse and its possible knowledge of a co-conversant's conflicting model. One area of future research in my work is designing rules, representations, and mechanisms for dealing with conflicting discourse modules.

KL-ONE Interface

Discourse engagement is an interactive process. As a result of discussion, one will often amend one's long term conceptualization of a co-conversant and/or one's personal beliefs about the world. Additionally, engaging in discussion often requires positing temporary, alternate views of the world in order to capture and respond to a co-conversant's statements.

Major issues which I am interested in are (1) representing and dealing with definitions of terms that differ between two conversants for what seems to be a same concept; and (2) delineating possible structural criteria by which to decide how these alternate definitions affect the definitions of other related concepts in the network.

For example, in informal debate, a conversant may challenge another conversant's assertion by challenging one of the consequences of this assertion. For instance, if someone asserts that a "bird is a type of poodle," we can challenge the assertion by claiming that "birds do not bark."

The simplest and most consistent use of KL-ONE here, I believe, would be to integrate the concept of a bird being a poodle into a somewhat modified version of one's initial view of the world. For example, here, the second version would change an exclusive subset relation between birds, dogs, and mammals, but it would retain the knowledge that dogs do bark. Then, using basic features of KL-ONE, like inheritance, we could derive the above type of challenge.

Groups like the BBN natural language group plan to represent assertions as propositions in such a way that the propositional content does not affect the initial knowledge of the system. So, for example, a conversant's assertion that a bird is a type of poodle would not affect a hierarchical pre-existing relationships between birds, dogs, and mammals in the description world, nor would it cause creation of a second description world containing such modification.

In addition, given the pre-existing definition that birds and dogs are mutually exclusive subsets of mammals, it seems that it would be "illegal" to create such a conflicting description in KL-ONE. In my opinion, if it is illegal, then, at maximum, it should be illegal to integrate the description into one's own personal description space that has this exclusive subset relation. However, once we can begin to index and partition the description world so as to distinguish between one's own definitions and the seeming definitions of a co-conversant, such an integration should not be illegal.

At the Workshop, then, I would be interested in learning the mechanisms currently being considered for using inheritance inferences and the like for propositions, discussing possible alternative mechanisms (like directly representing the propositional content into a definition space), discussing methods for deciding what parts of an initial knowledge space should be retained in a modified version, and, fourthly, discussing the issue of indexing and distinguishing between possible worlds so that we can use similar mechanisms and representations to reason about one's own personal knowledge and what seems to be a co-conversant's knowledge space.

Lastly, given the above example, while the description for a bird being a type of poodle would only be in one's KL-ONE description world of one's co-conversant, the actual assertion of this utterance clearly lies in one's own discourse world as well. Thus, in modeling conversations, we will need to monitor discourse worlds in addition to separate description and belief spaces.

Ethan Scarl, Carl Engelman, Tom Fawcett, Mike Pazzani,
Frank Jernigan, Bud Frawley, Peter Andreae, Sheila Sundaram

The Knowledge-Based Systems Group,
MITRE Corporation, Bedford, MA

This describes our data representation and related topics from a perspective outside of the KL-ONE experience, since this workshop represents our initial close encounter with KL-ONE.

KNOBS

MITRE's Knowledge Based System (KNOBS) is an experimental system which displays mixed initiative in the completion and modification of stereotypical plans and provides friendly truth maintenance and choice generation. Among other facilities, the system responds to questions about the database posed in English and offers the use of restricted English for the alteration of the system's behavior through the modification of its production rule logic.

KNOBS-FRL

KNOBS now maintains a database, which includes plan and planning template representation, embedded in a translation (into INTERLISP) and extension of MIT's Frame Representation Language (FRL). In FRL, the attributes (slots) of objects (frames) are related through facets to values and other associated information. The facet corresponding to "ordinary" values ($VALUE) can be supplemented by other system-defined facets which contain default information ($DEFAULT), procedural attachment for forward-chaining ($IF-ADDED and $IF-REMOVED), backward-chaining ($IF-NEEDED), or type-checking of values ($REQUIRE). Our experience is similar to others in finding $DEFAULTs to be redundant, since default information is usually represented by the inheritance of values through a generalization hierarchy.

FRL's generalization hierarchy has been extended to allow inheritance along arbitrary attributes (called "colors"), under user control. We have attempted to integrate conceptual individuation into the language by defining an "AIO" (An-Individual-Of) slot to indicate concrete objects. The default colors for inheritance are one (at most) AIO followed by any number of AKO links.

AIO links have been traditionally interpreted as designating set membership and AKO as subset inclusion, but there are problems with this. Since the notion of set membership has been superimposed upon that of individuation (as has the notion of set been superimposed upon that of generic concept), we have ended up using another pointer type (with programmer-defined semantics) to indicate set membership of sets within other sets.

Templates

We have introduced another class of frames, called "templates", which contain procedural knowledge for planning and constraint management. A template is attached to a generic frame, and controls individuation in that its attributes correspond one-to-one with those expected in the individual. The "values" of template slots contain procedures for calculating or requesting the values for the individual. The template thus appears closer in intent to a KL-ONE description than do our generic frames.

This scheme separates instructions for individuation from attribute inheritance. Structurally, of course, templates are inelegant compared to KL-ONE descriptions: if one slot is to have a "value-restriction" requiring it to be filled by some particular sort of frame, then the attached procedure must return that frame, and the constraints (below) must verify its appropriateness. We could start using the generic frames as more of a structural description for individuals. We have one active form of "modality": some template slots are marked "optional", which means that they are not instantiated in the individual unless specifically requested by something (typically the user, or a failing constraint).

Constraints

The original type-checking facility of FRL (the $REQUIRE facet) has been replaced by a more general CONSTRAINTS attribute which resides in the template and contains references to constraining conditions among the slot values (in the individual) of any subset of attributes. These constraints check the validity of the frames' values, their mutual consistency, and their appropriateness to the individual being assembled. Again, they are structurally less elegant than KL-ONE's RoleValuemaps since their functionality is procedurally determined. However, the constraint references contained in the template are themselves a useful sort of structural description-- they contain the knowledge that a constraint exists between some set of slots (possibly outside the concept being individuated). The structural usefulness of these constraint references may be diminished our current restriction that the names of the referenced constraints be unique, but they could be given inheritance capabilities which distinguish ordering relations from equivalence classes, etc.

Integrated Inference Architecture

We have avoided some of the common problems in representing universal restrictions among objects and quantificational and negative information, by using constraints as a reservoir of procedural information. On the other hand, some constraints are implemented as production rule invocations in an attempt to avoid some of the problems common to purely procedural knowledge bases. The FRL database and the production rule system form a hybrid system in which rule invocations and database accesses may trigger each other.

Finally

Many of our representational problems remain unsolved, of course, and many are still being discovered. We need, for example, to deal with abstract or vague restrictions on values input by the user, and which may apply to objects not yet created or to attribute values not yet filled.

We look to the KL-ONE community for sympathy and ideas.

Stuart C. Shapiro, Donald P. McKay, Joao Martins,
Ernesto Morgado, Michael Almeida, Han Yong You,
Gerard Donlon, Jeannette Neal, Terry Nuter, Lyn Tranchell

SNeRG, The SNePS Research Group,
Department of Computer Science,
State University of New York at Buffalo

SNePS, the Semantic Network Processing System [3], is an alternative knowledge representation system to KL-ONE. However, we find maintaining communication with the KL-ONE community useful because both our groups are interested in carefully thought out, fundamental issues of knowledge representation. On the other hand, KL-ONE and SNePS are also complementary in that KL-ONE is an "epistemological level" network while SNePS is a "logical level" network (see [1]). KL-ONE's structural primitives are at a higher level than SNePS', deriving from an analysis of how concepts are structured. SNePS' primitives derive from a version of predicate logic. The advantage of KL-ONE's level is that its networks express epistemological notions clearly and that epistemological level inferences should be fast since they are implemented directly. The advantages of SNePS' level are that different epistemological structures can be tried easily, since SNePS makes no commitments at that level, and that the semantics of SNePS' primitives are easily understood, since they are based on predicate calculus.

To further demonstrate the feasibility of using SNePS to experiment with various epistemological level representations, and to demonstrate the explication of the semantics of an epistemological level network in a logical level network, Lynn Tranchell is currently implementing KL-ONE in SNePS [5]. Essentially, this requires expressing the KL-ONE inheritance rules explicitly as deduction rules in SNePS. SNePS path-based inference [2, 4] may be enough for this although node-based inference is also available. Path-based inference infers an arc relation between two nodes from the presence of a specified path of arcs between the same two nodes. Node-based inference in SNePS uses rules expressed in the network using a formalism as rich as predicate calculus. Path-based inference uses rules expressed as sentences over a regular grammar of arc relations.

One thing we have found is that existing documentation of KL-ONE does not supply clear statements of KL-ONE's inheritance rules. With Ron Brachman's help, we have now formulated some of these rules, a sampling of which follows:

```
SUPERC <= (KSTAR SUPERC)

ROLED <= (RELATIVE-COMPLEMENT
                (COMPOSE (KSTAR SUPERC) ROLED)
                (COMPOSE (KSTAR SUPERC)
                         ROLED
                         (KPLUS MODS)))

STRUCTURE <= (RELATIVE-COMPLEMENT
              (COMPOSE (KSTAR SUPERC) STRUCTURE)
              (COMPOSE (KSTAR SUPERC)
                       STRUCTURE
                       (KPLUS PREEMPTS)))

NAME <= (COMPOSE (KSTAR  (OR DIFFS MODS)) NAME)

V/R <= (COMPOSE (KSTAR  (OR DIFFS MODS)) V/R)

INDIVIDUATES <= (COMPOSE INDIVIDUATES (KSTAR SUPERC))
```

Tranchell's implementation of KL-ONE will also include an implementation of JARGON as the KL-ONE user's language, although here too, the absence of a clear manual is hindering us.

REFERENCES

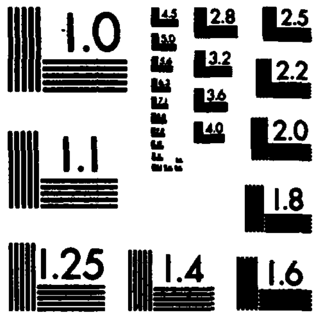
[1] Brachman, R. J. (1979)
"On the epistemological status of semantic networks." In Findler, N. V. (ed.), Associative Networks: Representation and Use of Knowledge by Computers. Academic Press, New York, pp. 3-50.

[2] Shapiro, S. C. (1978)
"Path-based and node-based inference in semantic networks." In Waltz, D. (ed.), TINLAP-2: Theoretical Issues in Natural Language Processing. ACM, New York, pp. 219-225.

AD-A123 772   PROCEEDINGS OF THE 1981 KL-ONE WORKSHOP(U) BOLT BERANEK AND NEWMAN INC CAMBRIDGE MA   J G SCHMOLZE ET AL. JUN 82 BBN-4842 N00014-77-C-0378   3/3

UNCLASSIFIED   F/G 5/6   NL

END
DATE
FILMED
83
DTIC



MICROCOPY RESOLUTION TEST CHART
NATIONAL BUREAU OF STANDARDS-1963-A

[3] Shapiro, S. C. (1979)
"The SNePS semantic network processing system." In Findler, N.V. (ed.), Associative Networks: Representation and Use of Knowledge by Computers. Academic Press, New York, pp. 179-203.

[4] Srihari, R. (1981)
"Combining Path-based and Node-based Reasoning in SNePS." Technical Report No. 183, Department of Computer Science, SUNY at Buffalo, Amherst, New York.

[5] Tranchell, L. (in progress)
"A SNePS Implementation of KL-ONE," Department of Computer Science, SUNY at Buffalo, Amherst, New York.

Candace Sidner, Rusty Bobrow (co-authors),
David Israel, Jim Schmolze, Bill Woods,
Madeleine Bates, Brad Goodman

Knowledge Representation for Natural Language Understanding,
Bolt Beranek and Newman Inc.

SOME ISSUES RAISED BY REPRESENTING SPEAKER PLANS IN KL-ONE

An informal description of one of a user's tasks

We have been studying the KL-ONE structures needed to represent a speaker's plans. What we will present here is a description of one particular task we have investigated, and the type of constraints we would have to represent in KL-ONE to model adequately the various possible plans for this task. We have a formalization in mind for these constraints, using slight extensions to the RSR facility in KL-ONE, but we will not present it here due to space restrictions.

The task to be described here is one of the possible tasks which might be performed with the aid of KLONE-ED, a graphical KL-ONE editor with an English-language front end. It consists of introducing a new concept into a network to serve as a common abstraction of several sub-concepts of a given concept. Thus, for example, suppose a user wants to form a new class for, say, HOUSEPET from some of the subconcepts of ANIMAL.

We wish to represent a family of possible plans that the user might have to achieve this end. We wish to represent both the types of actions that might be involved in such plans, and the ordering constraints which must be placed on such actions. We assume that there is a concept TIME-ORDERED-PLAN in our plan network that represents the high-order abstraction a time-ordered plan. This concept has a role for Action, and an RSR specifying a partial-ordering of these roles to indicate time-constraints -- which actions must occur before which others. We wish to create a sub-concept CREATE-ABSTRACTION-PLAN of TIME-ORDERED-PLAN that describes the constraints on the actions involved in getting the KLONE-ED system to create such an abstraction.

What are the actions that the user must plan for in such a task? We assume that the user will have the system help in two aspects of the operation, the determination of which concepts to include under HOUSEPET and the actual reconstruction of the network and display.

Before the system will perform an action, it must have an idea both of the class of action, and the objects to be affected by the action. Thus, a part of a user's task is sometimes communicating about the objects that are part of an action. The objects must be singled out, and that task of singling out will require that the user have some method for establishing references to those objects. The task of establishing a reference consists of using a referential team or a deictic act that can be properly interpreted by the system (the user's conversational partner). Once understood by the system, the user can assume the system will use the same referential expressions (or associated pronominal terms) for those objects. Thus reference is an act that the user takes account of in his plans.

We assume that the user does not have in mind exactly the set of concepts to join under HOUSEPET, but instead has a general idea of what such concepts would have to be like. In such a case, the user would have to investigate each potential sub-concept (that is, the existing sub-concepts of ANIMAL) to see if it should be included as a HOUSEPET, and then, if it met his/her criteria, identify that concept to the system to be abstracted. Thus, the two classes of actions the user would have to take are to investigate a concept, and to identify it to the system as one of the concepts to be included under HOUSEPET.

To investigate the possible sub-concepts of HOUSEPET the user must get the system to provide information about the subCs of animal.

Let us assume the user either asked explicitly for the subCs or asked for them in a general way, and the system chose to provide this information by displaying the subCs, either all at once or by screenful. We assume that by inspection of the display, the user can determine which subCs s/he needs for the abstraction, and that s/he then must communicate which ones they are to the system. The user can say either "I need dog and cat" or "Remember dog and cat from their display; give me the next screenful. Remember turtle and bird." (This latter case is needed for multiple screens). Otherwise, the user could point to the subCs he wants. Both methods of establishing reference to the objects of interest for the abstraction will promote communication; which method is used may depend on personal preference about typing as opposed to pointing.

Note that the user does not have to find out what all the subCs are before he communicates to the system the ones he needs for the abstraction relation. Instead he can interleave the two acts. Thus this task shows a surprising feature which seems to be common to many plans -- while there are strict time constraints on certain subsequences of actions, these are still loose enough to allow the steps to be aggregated in a number of different ways. For example, in the event that there are many screenfuls of animal subCs, the user can look at the first screenful, point out or mention the ones he wants from that screenful and go on. Or given a menu, the user can point to a member of the menu and ask that it be drawn on the screen, and then repeat this process for each other screenful. The behavior has a tree structure like the one below:

```
            Create-Abstraction(HOUSEPET subCs)
                /                       \
               /                         \
        Investigate<subCs>          Point-out<subCs>
                  /    \_____________/____        \
             ___/          _________/    /         \
            /             /             /           \
           /             /             /             \
Investigate(FISH) Point-out(FISH) Investigate(BIRD) Point-out(BIRD)
```

We now come to the representation of the ordering constraints on such a plan. We wish to provide a minimal set of constraints, so that many variations on the plan fall under the basic one. In particular, we wish to cover both the case in which the user sequentially has each potential concept displayed alone on the screen, looks at it, then indicates yes or no to the system, and the case in which the user has the system display the concept ANIMAL and all of its sub-concepts on a screen, and then points at the ones s/he wishes to have included under "housepet".

Presumably, there are no constraints on the order in which concepts are investigated, and there are no necessary constraints on the order in which acceptable concepts are identified to the system. The only constraint is that, for each concept, it must be investigated <u>before</u> it is identified (if it is acceptable).

Thus, we need a means of expressing several aspects of the actions mentioned in the previous section; a means of stating the sequencing of actions over time as a partial order, a means of expressing the interleaving of actions, and of expressing the identity of arguments to different sub-sections.

Norman Sondheimer, Nathan Relles,
Giorgio Ingargiola (Temple University)

Online User Assistance,
Sperry Univac, Blue Bell, PA

Description:

The goal of this project is to design a structure and access method that can assist online users by explaining: 1) concepts relevant to the tasks they have to carry out and to the systems they use; 2) what operations can be performed, and when and how to perform them; and 3) errors, their meaning, causes, and remedies. This design is based on network knowledge representation schemes. Our typical application is an operating system command language. Each node represents a concept such as a command, parameter, operatio or abstract object type. These are all attached to descriptive text, i.e., we use textual as opposed to procedural attachment. The nodes are related in hierarchies of the typical sort, e.g., SUPERC. But in addition we are developing hierarchies specific to our problem domain. These include OPERATOR, MODE, SYNTAX, EXAMPLE, IS-ATTR, and ATTRIBUTE hierarchies using arcs with these labels.

KL-ONE:

The relationship with our effort is indirect. The goal of KL-ONE is to supply a rigorous foundation for knowledge representation. However, in limited domains more efficient storage structure may be available which, although not capable of making the same distinctions as KL-ONE, are sufficient for the task at hand. We believe that our area of online assistance is one such domain. The problem for us is that the relations we are defining need to have well-structured definitions. In an attempt to provide these, we are trying to define these new relationships in terms of KL-ONE networks. Hence these arcs are essentially macro arcs. As an example, consider the IS-ATTR arc which relates an "object" node to an "attribute" node. This attribute node is related to other object nodes through a COMP relation. These object nodes show the types of objects that would be distinguished by differing values for the attribute. For example, the File node could IS-ATTR a

Residence attribute which could COMP a Disk File node and a Tape File node. Hence the IS-ATTR and COMP relations are capturing what would otherwise be captured by SUPERC, DATTR, and RESTRICTS relations. What we are doing is to define the IS-ATTR and COMP relationships in terms of these more elementary relationships.

Norman Sondheimer, Ralph Weischedel,
Stan Kwasny, Amir Razi, Sudhi Advani

Natural Language Processing Systems and Ill-Formed Input, University of Delaware/Sperry Univac

The goal of our project[1] is to add robustness to natural language understanding systems by adding rule-based methods of handling ill-formed input [1]. This includes both ungrammatical and semantically inappropriate utterances. The method being used consists of the following procedure:

1. attempt to process input as if it were well-formed,

2. if it cannot be understood as well-formed, localize the area of the suspected problem, and

3. based on what was found in attempting to process the input as well-formed, execute one or more "meta-rules" that express both how to relax some unsatisfied constraints and how to resume processing.

The meta-rules essentially characterize the types of errors that people can make. They are "meta" because they state how to rewrite the rules of normal processing to accept the ill- formedness.

To date, the majority of our work has been on the problem of ungrammatical input using the ATN formalism. We have been working with the RUS grammar, developing an interpreter for it, and adding meta-rules to it.

We have begun work on semantic ill-formedness by focusing on the area of case frame matching. In order to give form to our work, we have been considering the storage of case frames in the KL-ONE representation and the case matching algorithm employed in the PSI-KLONE interface [2]. The

[1]Natural Language Processing Systems and Ill-Formed Input (University of Delaware/Sperry Univac; work partially supported by the National Science Foundation under Grant No. IST-8009673)

Incremental Description Refinement (IDR) algorithm sequentially restricts the set of case frames under consideration as new syntactic units are interpreted. Consider Figure 1, which shows a part of the hierarchy of structures for clauses with "create" as their head. The clauses are differentiated by the types of LSUBJ (logical subjects) and LOBJ (logical objects) they accept. On input of "Schmidt created X.LSP", the IDR algorithm would restrict focus to nodes identified as 1, 2 and finally 3 as the verb, the subject and finally the object are identified.

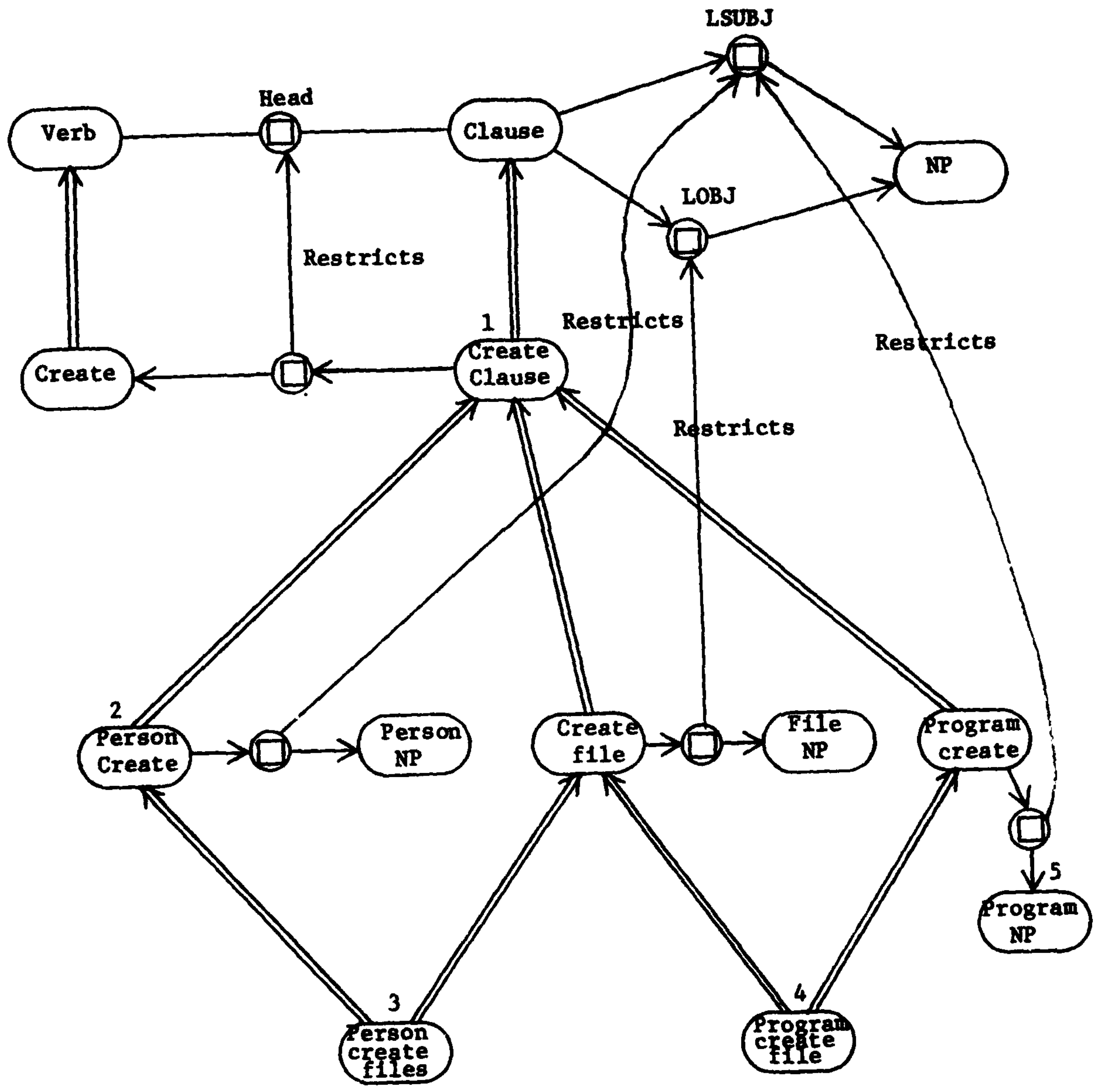
In order to focus on our work, consider the ill-formed input "X.LSP created a file". According to the KL-ONE hierarchy, files can not create files. However, our goal is to develop methods to give each input the best possible partial interpretation.

We first localize the problem. One of several promising heuristics is to consider only the nodes where blocks occurred and the largest number of constituents were accounted for. These would be nodes 3 and 4.

We then apply our meta-rules for semantic relaxation. The basic form of the rules is shown in Figure 2. We require that all conditions be matched before any and all actions are applied. The conditions test the state of the interpretation, including the input string, at the time of the blockage. The actions show how to relax the constraints and restart the interpretation. An appropriate meta-rule for our example is given in Figure 3. It effectively states that a possible semantic error is to use a file to refer to a program. This rule would apply to change node 5 which is associated by inheritance with the blockage at node 4. This relaxation then allows the modified node 4 to be seen as the best interpretation possible for the ill-formed input.

The advantages we have gained by working with KL-ONE are in its well-structured and well-defined taxonomies that make it possible to represent the case and selectional restriction hierarchies cleanly. This rigor is essential as we continue to develop our meta-rules.

FIG. 1. SAMPLE PSI-KLONE STYLE HIERARCHY



```
                 META-RULE STRUCTURES

                 C1 C2 ... Cn  => A1 A2 ... Am
      Ci :  Conditions
               (Case-Failure (test concept) head case)
                 ---assigning the constituent as the case of
                       head failed because (test concept)
                       failed.
              (Semantic-Class test)
                 ---the predicate given by test is a semantic
                       class.

      Ai :  Actions
              (New-Configuration substitution1 substitution2 ... )
                 ---make the given substitutions in the case
                       matching configuration.

                      Substitutions
                         (Substitute-in-Case  pattern1  pattern2 )
                            ---use pattern2 instead of pattern1 in
                                  case matching.
                         (Failed-Constraint string)
                            ---add the given string as a deviance
                                  note to the interpretation.

              (Resume) ---resume case matching.
```

FIG. 2. THE FORM OF META-RULES AND EXAMPLE CONDITIONS AND ACTIONS.

```
(CASE-FAILURE? (PROGRAM ?Z) ?Verb ?Case)

---> (NEW-CONFIGURATION
        (FAILED-CONSTRAINT "FILE REFERS TO PROGRAM")
        (SUBSTITUTE IN-CASE (PROGRAM ?Z) FILE ?Z)))
     (RESUME)
```

FIG. 3. META-RULE TO RELAX RESTRICTIONS SO THAT REFERENCES TO PROGRAMS CAN BE MADE THROUGH REFERENCE TO FILES

REFERENCES

[1] R. M. Weischedel and N. K. Sondheimer (1981) "A Framework for Processing Ill-Formed Input," Newark, DE: Dept. of Computer and Information Sciences, University of Delaware, and TM H-00519, Blue Bell, PA: Sperry Univac.

[2] R. Bobrow and B. L. Webber (1980) "Knowledge Representation for Syntactic/Semantic Processing," in Proceedings of The First Annual National Conference on Artificial Intelligence, 18-20 August 1980, Stanford, CA, pp. 316-323.

John Vittal, Rusty Bobrow, Oliver Selfridge

Naval Display and Projection in Tactical Situations, Bolt Beranek and Newman Inc.

There are several areas where we intend to use KL-ONE:

- o interface to AIPS
- o knowledge-based simulation
- o inferencing
- o constraints
- o representation of time-varying world models.

The knowledge-based simulation tool includes the other areas.

The use of an object-oriented language like KL-ONE is required for this type of simulation facility, which basically deals with a collection of objects and their relationships; KL-ONE is the only game in town. In a simulation model, many different types of objects must know different methods for updating their state when the simulation clock is "incremented". Flexibility derives from being able to extend the set of object categories dealt with by the system without invalidating procedural knowledge which has already been encoded. The database itself consists of know data (that is the current state of an object with varying likelihoods) and temporary data arrived at through the actual simulation process. The known data is updated periodically, but generally not during the execution time-span of a specific simulation "run". The use of this kind of facility are varied; the specific one which we are developing is a Naval Command and Control aid for use on-board ships.

The representation of time-varying models is crucial to such a simulation tool. The problem with representing these models is:

- o the relationship between time-variant and time-invariant attributes of the model

- the ability to represent this information concisely and efficiently
- applying a meta structure on sets of contexts

In addition, there is the problem of how to display the various contexts (i.e., internal states and state histories of decision models) in a user-oriented and reasonable fashion. The hope is that an interface to AIPS, which is implemented in KL-ONE, can be constructed for this purpose.

E. Judith Weiner
Computer and Information Sciences Department,
Temple University

USING KL-ONE TO UNDERSTAND METAPHORS

I am investigating the possibility of using KL-ONE in some natural language research which I am conducting. Specifically, I am working on metaphor processing although my work thus far has indicted that many other phenomena of language can be similarly processed.

In my approach, metaphors are treated as a class of utterances at the opposite end of a continuum from strictly literal speech. Rather than handling them uniquely then, they are processed by the same techniques that are used for literal language. And the same techniques used to understand classical metaphors of the sort John is a pig can also be applied to other figures of speech, such as personifications, oxymora, similes, ironies, hyperboles, and even certain categories of jokes.

The focus of this research is on the procedures necessary for computer "understanding" of natural language, whether literal or figurative. First, it is necessary to clarify, from a computational point of view, just what I mean by "understanding". The working definition which I am using is: A computer understands an input sentence correctly if it can produce an appropriate response (output sentence). This means that if I see fit to inform a machine that I'm dying of hunger (and it has also the information that it is noon on a normal working day, and I haven't been abandoned on a desert island), it will not prepare to erase my name from its files. It will rather integrate this information correctly and have it accessible when it needs to respond to me. In order to do this, it must have available to it some sort of a network of knowledge of the world which includes certain pragmatic considerations, such as time of day, identity of the speaker, etc. as well as some facts (roles) about hunger, dying, etc. In addition to the pragmatic considerations, a usable representation of knowledge must have the following characteristics:

1. There must be a store of general knowledge, in some accessible structure

2. The linguistic context (linear in nature) must be available to augment the store of general knowledge

3. The computer must be able to interrogate the user to clarify meaning if not available from general knowledge or context.

4. It must be possible to integrate user responses with existing knowledge, i.e., the computer should have the potential to learn (how and under what circumstances this is to be accomplished is, of course, no simple matter).

An example can illustrate how this can function. I say to the computer, "John is a pig." If this is the first thing I say, then all that the computer has to work with is its general knowledge. It doesn't know anything about John. It doesn't even know which John I am talking about. But it knows that John is a male name, it knows some general things about men and some general things about pigs. On this basis, it has two possibilities open to it. It can assume that John is a man's name (in which case it has to provide a metaphorical interpretation), or it can assume that John is a pig's name (in which case, no problem). It will have to get clarification from the user. But would anyone start an interaction with a total stranger (or a machine) in this way?

In normal discourse among strangers, a sentence like John is a pig would be preceded by considerable linguistic context which would clear up the above ambiguity. If this weren't the first interaction, (i.e., the conversants weren't strangers) the store of knowledge would have been considerably augmented by previous discussions. (One might, for example, know something about John's table manners.) So it should be between people and computers.

In the light of the above, these problems of processing figurative language are not substantially different from those of handling literal language. Literal language processing has to contend with disambiguation in general as well as resolving ambiguity of reference, e.g., John accompanied Jack to the restaurant. He was very hungry. and numerous other nasty problems such as ellipsis. The technique/theory which I am describing here is uniformly applicable to problems of both literal and figurative language. In this, it is a more general theory of processing than most. The corollary to this is that computational processing can function as a one-step operation instead of the commonly accepted technique involving two passes (one to recognize that a metaphor has been encountered; one to understand the metaphor). This is a plus in that it can result in a reduced expenditure of computer time.

I am considering a kind of global concept of a language understanding system in which the dynamic nature of knowledge such as I've described is taken into account. (How is a new piece of information added to the network? How does it affect the store of general knowledge? When is something a contradiction or a mistake rather than a metaphor - after all, John is not really a pig). How could KL-ONE figure in such a scheme?

If it is assumed that a semantic theory of natural language processing includes a KL-ONE type of knowledge network, then the following approach works nicely. Understanding a sentence involves a "reconciliation" of two (or more) portions of the network[1].

In the example, John is a pig, the KL-ONE diagram might appear as follows:

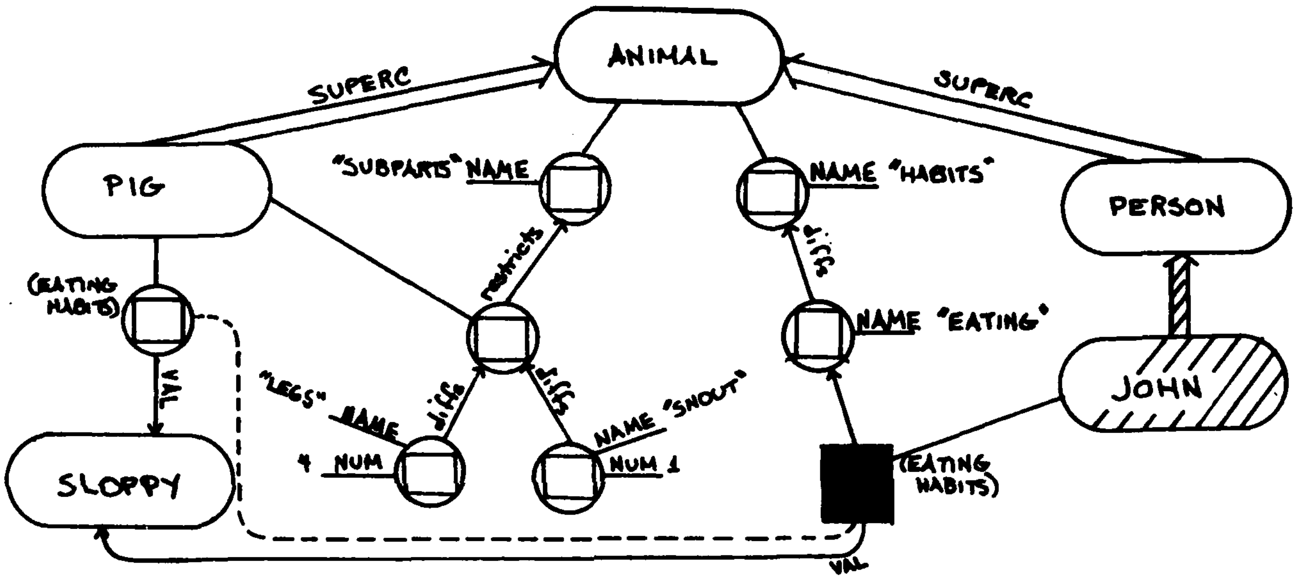


FIG. 1.

The result of processing the sentence might be the special link indicating the area that they have in common (indicated by the dotted line).

One of the main areas to investigate here is how such assertions can be represented. It seems that the line between assertions and definitions becomes blurred in examples such as the above (and this problem, of course, is

[1]There are similarities between my approach to John is a pig and Winston's Robbie is a fox. There are also many important differences, the discussion of which is beyond the scope of this position paper.

by no means limited to metaphors). Does the sentence, John is a pig contribute to what we consider to be the definition of John, or is this simply something we are asserting about him? If KL-ONE can be used to support a dynamic view of knowledge in which knowledge is acquired and modified on an ongoing basis, how can it decide to question for a given new processed sentence? I feel confident that the angle from which I am approaching the use of KL-ONE will not only raise some new issues and point out some new problems, but will provide some new solutions to old ones as well.

Bibliography

Carbonell, Jaime G. (1980)
"Metaphor - A Key to Extensible Semantic Analysis", Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics and Parasession on Topics in Interactive Discourse, 19-22 June 1980, Philadelphia, PA pp.17-21.

Kooij, Jan G. (1971)
Ambiguity in Natural Language: An Investigation of Certain Problems in its Linguistic Description. Amsterdam: North-Holland Publishing Company.

Lakoff, George and Mark Johnson (1980)
Metaphors We Live By. Chicago: The University of Chicago Press.

Slack, Jon M. (1980)
"Metaphor Comprehension - A Special Mode of Language Processing?", Proceedings of the 18th Annual Meeting of the Association for Computational Linguistics and Parasession on Topics in Interactive Discourse, 19-22 June 1980, Philadelphia, PA, pp.23-24.

Winston, Patrick H. (1979).
"Learning by creating and justifying transfer frames." In Patrick Winston and Richard H. Brown (Eds.), Artificial Intelligence: An MIT Perspective. Cambridge, MA: The MIT Press, Vol.1, pp.349-374.

Ralph Weischedel, Daniel Chester, Robert Vollum

Department of Computer and Information Sciences
University of Delaware

Prerequisites to Deriving Formal Specifications from Natural Language Requirements

We are implementing a system which takes an English description of a data structure and creates a formal specification for it as an abstract data type[1].

For example, our system will handle texts such as the following one, which was constructed from pages 77 and 79 of Horowitz and Sahni [3].

> A stack is an ordered list in which all insertions and deletions are made at one end called the top. Given a stack S=(A[1], ..., A[N]) then we say that A[1] is the bottommost element and A[I] is on top of element A[I-1], 1<I<=N. Associated with the object stack are several operations that are necessary:
>
> - CREATE(S) which creates S as an empty stack;
> - ADD(I,S) which inserts the element I onto the stack S and returns the new stack;
> - DELETE(S) which removes the top element of stack S and returns the new stack;
> - TOP(S) which returns the top element of stack S;

[1]Acknowledgement: Research sponsored by the Air Force Office of Scientific Research, Air Force Systems Command, USAF, under grant No. AFOSR-80-0190. The United States Government is authorized to reproduce and distribute reprints for Governmental purposes notwithstanding any copyright notation herein.

o ISEMTS(S) which returns true if S is empty else false.

The output will be a specification in first order logic with primitive terms for sets, vectors, records, integers, reals, and characters. (Such an output is very close to an existing specification language [5].)

We are paper users of KL-ONE, in that we use it to help organize our approach to knowledge representation. One use is in the organization of case frame semantics for semantic interpretation of English. The syntactic component of our system is the RUS grammar [1]. As the parser finds complete constituents, it sends messages to a semantic component for interpretation and possible veto of the partial parse. Therefore, the case frames must be organized to interpret constituents incrementally and to identify impossible descriptions in the application of data structure specification. The relevance of KL-ONE for this task has been demonstrated in Bobrow and Webber [2].

A second use arises from the fact that the person's description of a data structure tends to be quite far from the concepts available in formal specification languages. For instance, people tend to describe stacks in spatial terms (e.g. "top", "bottom", "position", etc.). Formal specifications involve mathematical objects, e.g. sets, tuples, functions, records, etc. This gap seems pervasive in descriptions of data structures. KL-ONE's strength for this problem of mapping between two rather distinct conceptual structures is its precise, semantically clear means of organizing concepts and their attributes. Thus, it can organize the two different sets of concepts. Furthermore, the techniques of Mark [4] for mapping between KL-ONE concepts could be appropriate for our project.

Areas where KL-ONE could be improved for applications such as ours are in roleset relations and the addition of a powerful inference mechanism which is combined with a reasoning maintenance system. As an example of some of the concepts one wants to represent at the level of a formal specification, consider the KL-ONE net in Figure 1. To define functions as a specialization of the concept of set, one must state at least the following:

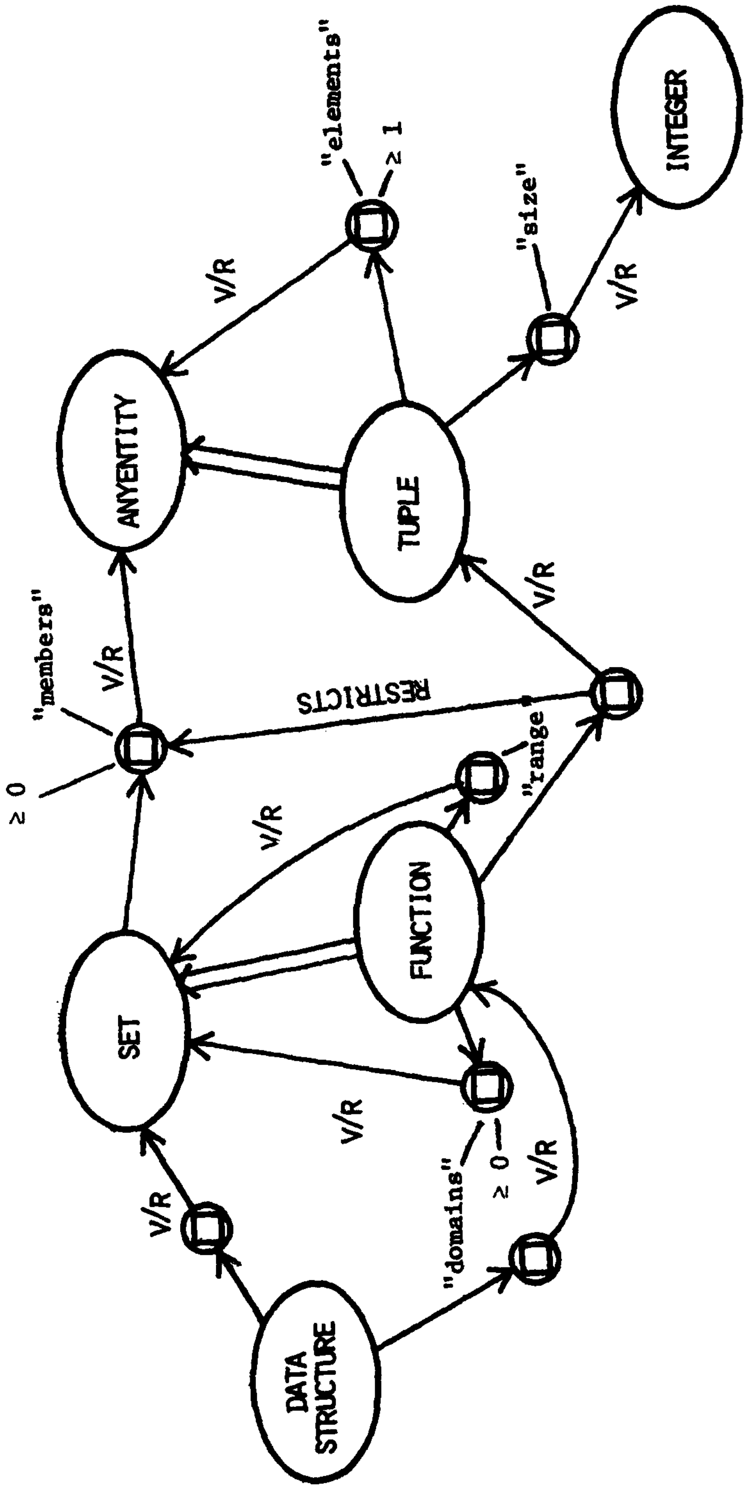
1. All of the tuples that make up a function must be the same size.

2. Let us call the size of the tuples n. For 1<=i<n, there is a domain such that for all tuples the element in the ith position of the tuple comes from that domain set.

3. All elements in the nth position of some tuple come from the range set.

4. It is impossible to have two tuples which are identical on their first n-1 elements, but differ on the nth.

FIG. 1.


INTEGER
"elements"
≥ 1
"size"
V/R
V/R
ANYENTITY
TUPLE
V/R
"members"
V/R
RESTRICTS
"range"
≥ 0
V/R
FUNCTION
SET
V/R
V/R
"domains"
≥ 0
V/R
DATA STRUCTURE

Consequently, one should be able to state conveniently that a particular roleset is a list and be able to refer conveniently to the ith element of the list (as well as the first and remaining ones). Additionally, one must be able to express conveniently at least what can be stated easily in first order logic.

REFERENCES

[1] Bobrow, Robert J. (1978)
"The RUS System", in Research in Natural Language Understanding, by B. L. Webber and R. Bobrow, BBN Report No. 3878, Bolt Beranek and Newman, Inc., Cambridge, MA.

[2] Bobrow, Robert J., and B. L. Webber (1980)
"Knowledge Representation for Syntactic/Semantic Processing," in _Proceedings of The First Annual National Conference on Artificial Intelligence_, pp. 316-323.

[3] Horowitz, Ellis and Sartaj Sahni (1976)
Fundamentals of Data Structures, Computer Science Press, Inc., Woodland Hills, CA.

[4] Mark, William (1980)
"Rule-Based Inference in Large Knowledge Bases", in _Proceedings of The First Annual National Conference on Artificial Intelligence_, pp. 190-194.

[5] Roubine, Olivier and Lawrence Robinson (1976)
SPECIAL Reference Manual, Technical Report CSG-45, Stanford Research Institute, Menlo Park, CA, August.

Gerry Wilson, Jane Barnett

VIEW Project,
Computer Corporation of America

KNOWLEDGE REPRESENTATION, KL-ONE, AND THE VIEW PROJECT

The VIEW system is intended to be a high level user interface for information systems. VIEW is designed to use a separate (physically independent) data base management system to handle data values (instances). The types of data available cover a broad range, including characters, text, numbers, pictures, diagrams, combinations of these, and ultimately sound.

To the user the data appears to be arranged spatially, a la the approach employed in the Spatial Data Management System. This means that the data is arranged on a collection of two dimensional planes, which are in turn linked into a network. The user passes from one plane to another by means of bi-directional ports, each of which connects a specific region of one plan to a specific region of another. As a user enters a port to pass to a lower level plane (called zooming in) the effect is to enlarge the perspective by giving more detailed data about that segment of the total information space. Returning back through a port traversed earlier (called zooming out) provides a more global and less detailed perspective.

There are three logical portions of the knowledge base in VIEW:

1. a meta-level description of the information available to the user,

2. a semantic description of pictures in terms of the underlying database, and

3. a rule-based collection of graphics composition knowledge.

The meta-level description of the available information is used to determine the segment of the available data that is relevant and useful to the user asking the question, and appropriate clusters and subcluster of both objects and their attributes. This information together with the picture semantics and graphics composition knowledge is used to determine the arrangement of objects and attributes across and within the collection of two-dimensional planes, the icons to be used to represent the data to the user, and the appearance of these icons.

VIEW is principally a user of knowledge, rather than a creator. Although we would expect this to change in the future, VIEW does not alter its knowledge base as a result of the user actions. The structure and content of the knowledge base is controlled by a human "VIEW Administrator".

Users pose questions to VIEW by means of a menu facility driven by the content of the knowledge base. Thus the users explicitly select concepts and rolesets which are of interest to them. This initial cut at the answer space is refined by VIEW using the knowledge base. Concepts and rolesets which are expected to be of interest to the user, due to our prior knowledge of the user's perspective together with the general knowledge of the semantic structure of the data base, may be added to the "answer space". The resulting information constituting the answer space is then further structured to allow it to be presented as an appropriate set of two dimensional planes. This requires consideration of both the semantic relationships and the graphical constraints of presentation. All of this knowledge is represented within our modified KL-ONE represented knowledge base.

The VIEW system is being designed and implemented in a multi-year project of which the first-pass will be completed this fall. To date our work with KL-ONE has been strictly on paper, although we are beginning to implement a primitive, first-cut of the knowledge base. Our plans call for having at least a subset of KL-ONE running early this fall.

In our application KL-ONE has several good points and several shortcomings. The good points are:

1. Epistemology = The approach of employing hierarchical collections of concepts and rolesets seems to capture the general nature of the object oriented worlds we have been using. Forcing us to structure this world carefully within the epistemological structure of KL-ONE is a clear win.

2. Inheritance = Being an object oriented world many important properties of interest to users posing questions to the database are inherited and/or derived from properties of higher level concepts.

3. Generic vs. instance distinctions = Due to our distinct split between meta-level knowledge and the specific data contained in the data base, having a corresponding split in the KL-ONE representation is most important.

The shortcomings we have found with KL-ONE in our application are:

1. Handling of independent data instances = Because our approach employs a data base of concept and roleset instances which is distinct from the knowledge base, it is essential that we be able to capture the instance references in the knowledge base without the specific data values. Our current approach is to add a new type of node, called a logical data base reference node. This node falls somewhere between KL-ONE's individual and generic concept nodes. It describes a set of specific data values, where the data may be found in the data base, and how to query the DBMS to obtain the data values.

2. Alternative perspectives = The knowledge base must capture the meta-level description of the data base applicable to all the potential users of the information system. Yet clearly different users, and even the same user at different times, have somewhat different perspectives on the relative importance of various concepts and rolesets, and may even perceive a different organization of the data collection. We need to be able to capture these differences without providing multiple independent knowledge bases and without requiring computationally costly modifications to temporary copies. Our current approach has been to modify the nature of the structural description nodes to incorporate predicate logic descriptions of concept structures and associated bond-strength information specifying the semantic strength of bonds between concepts and rolesets.

3. Search efficiency = Although it is conceptually important to capture the relationships between various rolesets and lower level concepts by means of the proper subset, roleset, instance, restriction, etc. types of arcs, this does not always lead to efficient search abilities. In particular, it is possible to systematically define some new links which represent contractions of longer paths thus reducing the traversal times. While we do not yet fully understand all the implications of making such changes in the KL-ONE representation, we believe that changes like this are essential for our application. If searches of the network are too costly, our user interface is doomed to failure because its response will be too slow.

4. Heuristic knowledge = In our presentations there are many alternatives available for the arrangement of the data both within each plan and across planes, and for choices of scale, icons, colors, etc. The selections must be made using a variety of heuristic guidance measures, some of which are global and some of which are local, that is, specific to the particular data to be presented. Thus, it is important for us to have a way to represent this sort of meta-meta data within the uniform knowledge

representation structure. We are developing special nodes and arcs to handle this type of information.

It is also of interest to note that our work is being done on a VAX and that therefore we are involved in the problems of porting at least subsets of KL-ONE to the VAX environment (currently to Franz Lisp).

Beverly Woolf
Department of Computer and Information Science,
University of Massachusetts at Amherst

AN INTELLIGENT TUTOR FOR BEGINNING PROGRAMMERS

We have implemented a prototype bug finder/tutor which detects run-time semantic errors in student programs. It has been successfully tested in a class of over 800 students where it described errors, provided tutoring advice, and saved a copy of its results.

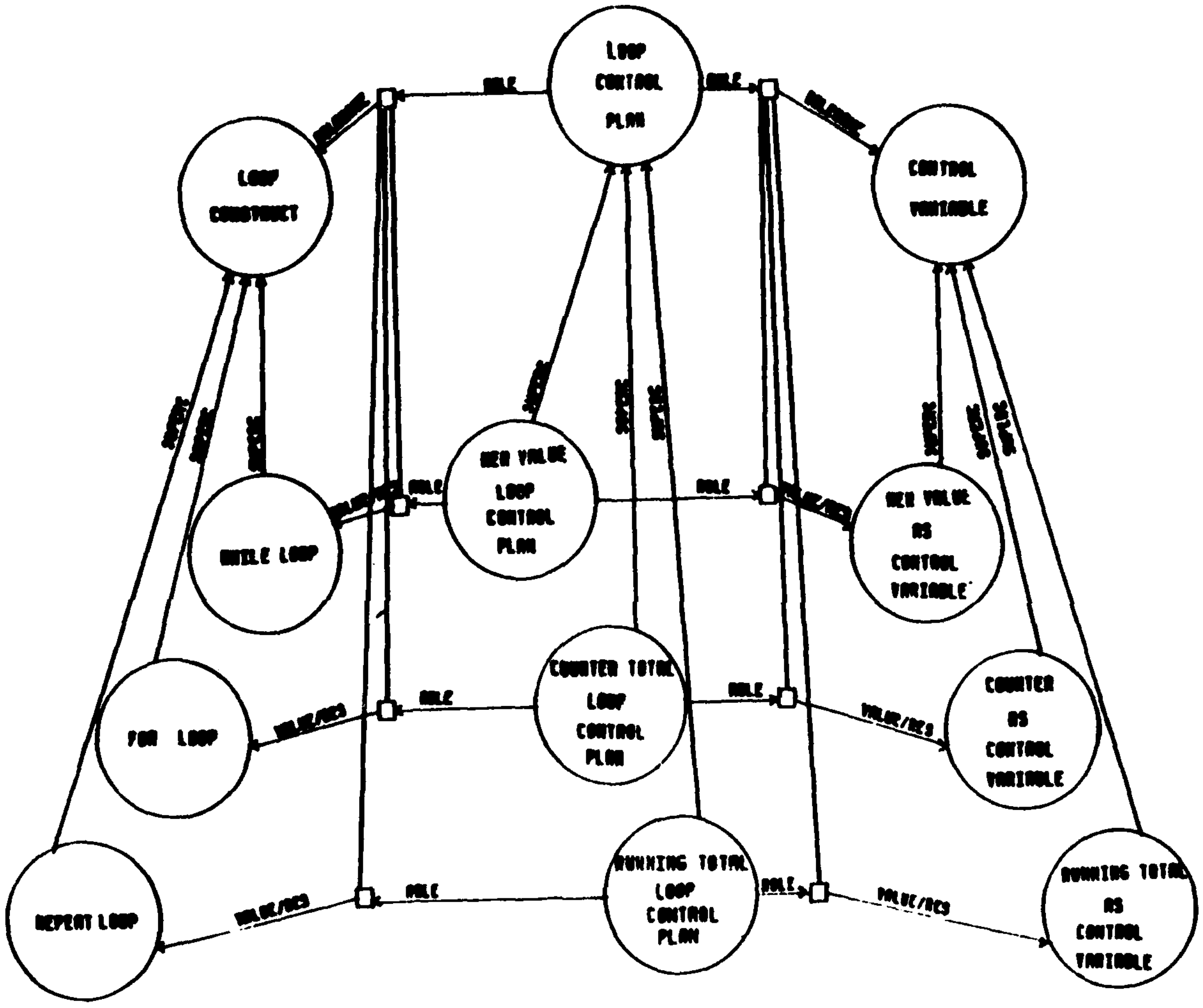
A tutor which will dynamically generate explanations and tutoring advice to students is being developed in LISP on a VAX 11/780 using a UMass implementation of the basic elements of KL-ONE. It uses empirical studies in cognitive processes of programming to identify templates that experts use when translating problem descriptions into code. These high level procedural plans are the basis for representing programs in the KL-ONE network. Both correct and incorrect code is represented as instantiations of the high level plans. The knowledge base now contains 300 nodes, or about half of the concepts in a one semester course in PASCAL, including WHILE, FOR, and REPEAT loops, variable roles, running sums, counter totals, and assignment statements.

We frequently generate our explanations of the correct code in Pascal after describing a student's incorrect code. This requires repeated movement between the correct and the incorrect data bases. We needed a link to speed up this process and to reflect our pragmatic requirements independent of domain knowledge. As a computational shortcut, we built "meta-links" between buggy knowledge and the corresponding correct knowledge.

REFERENCES

[1] Bonar, J., Ehrlich, K., Soloway, E., and Rubin, E. (1981) "Collecting and analyzing on-line protocols from Novice Programmers", Technical Report 81-12, Department of Computer and Information Science, University of Massachusetts, Amherst.

[2] Soloway, E., Woolf, B., Rubin, E., and Barth, P. (1981) "Meno-II: An Intelligent Tutoring System for Novice Programmers", proceedings of IJCAI-81, Vancouver, B.C.


LOOP CONTROL PLAN
ROLE
ROLE
LOOP CONSTRUCT
CONTROL VARIABLE
SUPERC
SUPERC
SUPERC
NEW VALUE LOOP CONTROL PLAN
ROLE
ROLE
VALUE/DES
WHILE LOOP
NEW VALUE AS CONTROL VARIABLE
COUNTER TOTAL LOOP CONTROL PLAN
ROLE
ROLE
VALUE/DES
VALUE/DES
FOR LOOP
COUNTER AS CONTROL VARIABLE
RUNNING TOTAL LOOP CONTROL PLAN
ROLE
ROLE
VALUE/DES
VALUE/DES
REPEAT LOOP
RUNNING TOTAL AS CONTROL VARIABLE

Frank Zdybel, Norton Greenfeld,
Jeff Gibbons, and Martin Yonke

Advanced Information Presentation System (AIPS) project,
Bolt Beranek and Newman Inc.

In our work with AIPS, we have found a number of things about KL-ONE which give us trouble:

- It is generally hard to represent the behavior of a knowledge-based system written in KL-ONE because the implementation is incomplete with respect to RSR's and SD's (no enforcement or checking built in, no breadth-first ordered inheritance of SD's.) Even if enforcement should at this stage be left to the user, there should be some hooks (other than IHooks) to allow clean implementation.

- It is difficult to express some types of domain relationships because an SD can only pertain among the Roles of a single Generic Concept. We feel we need to be able to describe domain relationships as pertaining among the roles of objects, and not strictly either as pertaining among objects or among the roles of a single object. In doing this, we would like to have the fully declarative SD mechanism at our disposal.

- The BBN Interlisp implementation is difficult to use because it lacks a comprehensive and readable print representation, a reader/printer with incremental, file-oriented dumping capability, and an in-core editor.

- As we attempt to use less familiar features of the language, the lack of a good primer/reference manual is troublesome.

KL-ONE features we don't like:

- Saved lists and IHooks make the code of the Interlisp implementation ferociously complex. They make it difficult to determine the exact semantics of KL-ONE functions by reading the implementation, and they make it more difficult to modify or extend the language to suit special requirements. We believe the Saved inheritance results are a large factor in KL-ONE's unseemly appetite for memory.

- In our view, IHooks are an extremely debatable feature because they complicate debugging. However, they are a potentially valuable escape mechanism. Realistically they should be viewed purely as a method of advising KL-ONE functions per specific sites in a network. If IHooks are to be kept for this purpose, we believe the OVERRIDE option should be restored. IHooks cannot provide an adequate declarative description of a default, and for this reason we believe the DEFAULT option should be dropped.

What do we like?

- Generally speaking, we are happy with the purely descriptive syntax of the language (Concepts, Roles, Wires and Cables).

- We are happy with a philosophy of incremental implementation of the KL-ONE notation because there are good escapes to LISP (tags and itags) and because of the metadescription feature of KL-ONE. These have made it possible for us to "get along" with the language.

APPENDIX A
AGENDAS

A.1 Technical Discussions

<u>Friday, October 16th:</u> <u>morning and afternoon</u>

Assertions, etc.	
representing propositions and asserting them in contexts	
coreference	
"contingent superCs"	Ron Brachman
defaults	and
Contexts/possible worlds	Hector Levesque
object constancy	
variables	
Realization.	Bill Mark

<u>Friday evening</u>

System utilities	Jim Schmolze and Frank Zdybel

<u>Saturday, October 17th:</u> <u>morning</u>

Structural Relations:	
RSRs	
sequences & ordering	Rusty Bobrow
quantification in RSRs	
QUA.	Mike Freeman

<u>Saturday evening</u>

Individual Concepts.	Bill Mark
Classifier	Tom Lipkis

Sunday, October 18th: morning

Individual Concepts (cont'd)	Bill Mark
System maintenance	Jim Schmolze
Classifier (cont'd).	Tom Lipkis

A.2 General Conference

Sunday, October 18th

6:30p -- Hors d'oeuvres, cash bar in the conference room --
8:00p -- Buffet dinner in the dining room --

Monday, October 19th

9:00a Opening remarks
Jim Schmolze (BBN)
9:15a "Highlights from KloneTalk: Display-Based Editing and Browsing, Active Role Value Maps, Qua Concepts, and Decompositions"
Richard Fikes (Xerox)
10:00a Review of technical discussions
Ron Brachman (Fairchild), William Mark (ISI), Jim Schmolze, Rusty Bobrow (BBN), Michael Freeman

10:30a -- Break --

11:00a Continuation of review of technical discussions

11:45a "Translating KL-ONE from Interlisp to FranzLisp"
Tim Finin (Penn)

12:15a -- Break for entire afternoon --

5:00p -- Coffee and cake --

5:30p "Towards a calculus of structural descriptions"
Michael Freeman (Burroughs)

6:00p Talks from various sites
1) "A Knowledge Based Graphical Interface for Databases"
Gerald Wilson (CCA)
2) "Between Description and Assertion"
Alan Frisch (Rochester)
3) Talks on 3 projects:
"Natural Language Processing Systems and Ill Formed Input"
"Prerequisites fo Deriving Formal Specifications from Natural Language Requirements"
"Online User Assistance"
Norman Sondheimer (Sperry) and
Ralph Weischedel (Delaware)
4) "The BBN Natural Language Understanding Project"
Candace Sidner (BBN)
5) "Extensible Semantic Networks"
Peter Patel-Schneider (Toronto)
6) "KL-ONE in SNEPS"
Stuart Shapiro (SUNY)

8:00p -- Dinner --

<u>Tuesday October 20</u>

9:00a "A knowledge representation model of prototype theory"
Benjamin Cohen (Xerox)

9:30a Group meetings - I
1) "Usage of contexts for representation of time and belief" Chaired by David Israel (BBN)
2) "Transporting KL-ONE to other Lisps"
Chaired by Tim Finin (Penn)

10:30a -- Break --

11:00a Group meetings - II
1) "Inferences in KL-ONE"
Chaired by Ron Brachman (Fairchild)
2) "Practice KL-ONE session"
Chaired by Rusty Bobrow and David Israel (BBN)

12:00 -- Lunch --

1:30p "A KL-ONE Classifier" - Tom Lipkis (ISI)

2:00p "A KL-ONE Based Knowledge Representation Kernel"
Ron Brachman (Fairchild)
2:30p Closing Remarks - Ron Brachman
2:45p Adjournment

APPENDIX B
PARTICIPANTS IN THE 1981 KL-ONE WORKSHOP

B.1 Technical Discussions

Rusty Bobrow (BBN)
Daniel Bobrow (Xerox)
Ron Brachman (Fairchild)
Richard Fikes (Xerox)
Michael Freeman (Burroughs)
Jeff Gibbons (BBN)
Austin Henderson (Xerox)
David Israel (BBN)
Hector Levesque (Fairchild)
Tom Lipkis (ISI)
William Mark (ISI)
Jim Schmolze (BBN)
William Woods (BBN)
Frank Zdybel (BBN)

B.2 Main Conference

Hassan Ait-Kaci (Penn)
Jane Barnett (CCA)
Madeleine Bates (BBN)
Ron Brachman (Fairchild)
Rusty Bobrow (BBN)
Daniel Bobrow (Xerox)
Amedeo Cappelli (Pisa)
Ben Cohen (Xerox)
Richard Duncan (Penn)
Richard Fikes (Xerox)
Tim Finin (Penn)
Michael Freeman (Burroughs)
Alan Frisch (Rochester)
Christopher Fry (MIT)

Jeff Gibbons (BBN)
Brad Goodman (BBN)
Norton Greenfeld (BBN)
Carole Hafner (General Motors)
Austin Henderson (Xerox)
David Israel (BBN)
Bryan Kramer (Toronto)
Bob Krovetz (NLM)
Henry Leitner (Boston University)
Hector Levesque (Fairchild)
Tom Lipkis (ISI)
William Mark (ISI)
Eric Mays (Penn)
David McDonald (UMass)
Lorenzo Moretti (Pisa)
Peter Patel-Schneider (Toronto)
Rachel Reichman (UCSD)
Nathan Relles (Sperry)
Ethan Scarl (MITRE Corp.)
Jim Schmolze (BBN)
Stuart Shapiro (SUNY-Buffalo)
Candace Sidner (BBN)
Norman Sondheimer (Sperry)
John Vittal (BBN)
Bonnie Webber (Penn)
Judy Weiner (Temple)
Ralph Weischedel (Delaware)
David Wilczynski (ISI)
Gerald Wilson (CCA)
William A. Woods (BBN)
Martin Yonke (BBN)
Frank Zdybel (BBN)

APPENDIX C
NAMES AND ADDRESSES OF MEMBERS OF THE KL-ONE COMMUNITY

Hassan Ait-Kaci
Dept. of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
33rd and Walnut Streets
Philadelphia, PA 19104
(ARPANET: AITKAC@WHARTON)

Professor James Allen
Department of Computer Science
Mathematical Sciences Building
University of Rochester
Rochester, NY 14627

Jane Barnett
Computer Corporation of America
575 Technology Square
Cambridge, MA 02139
(ARPANET: JANE@CCA)

G. Edward Barton
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139
(ARPANET: EB@MIT-AI)

Dr. Madeleine Bates
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: BATES@BBNG)

Robert J. Bechtal
Code 8242
Naval Ocean Systems Center
San Diego, CA 92152
(ARPANET: RBECHTAL@BBNA)

Dr. Daniel Bobrow
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94301

Robert J. Bobrow
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: RUSTY@BBNG)

Ken Bowen
CIS
313 Link Hall
Syracuse University
Syracuse, NY 13210
(ARPANET: BOWEN@USC-ECL)

Dr. Ronald J. Brachman
Artificial Intelligence Research Laboratory
Fairchild Research and Development
4001 Miranda Ave.
Palo Alto, CA 94304
(ARPANET: BRACHMAN@SRI-KL)

Professor David C. Brown
Department of Computer Science
Worcester Polytechnic Institute
Worcester, MA 01609

Dr. John Seely Brown
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(ARPANET: BROWN@PARC-MAXC)

Dr. Amedeo Cappelli
Instituto di Linguistica Computazionale
Via Della Faggiola 32
56100 Pisa
Italy

Professor Nicholas Cercone
Department of Computing Science
Simon Fraser University
Burnaby, British Columbia
Canada V5A 1S6

Eugene Ciccarelli
Artificial Intelligence Laboratory
Massachusetts Institute of Technology
545 Technology Square
Cambridge, MA 02139
(ARPANET: ECC@MIT-AI)

Ben Cohen
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(ARPANET: COHEN@PARC-MAXC)

Professor Philip R. Cohen
Department of Computer Science
Oregon State University
Corvallis, Oregon 97331

Jeff Conklin
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Jos de Bruin
Vakgroep Functieleer
Psychologisch Laboratorium
Weesperplein 8
Amsterdam

Richard Duncan
Dept. of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
33rd and Walnut Streets
Philadelphia, PA 19104
(ARPANET: DUNCAR@WHARTON)

Dr. Robert Engelmore
Computer Science Department
Margaret Jacks Hall
Stanford University
Stanford, CA 94305
(ARPANET: ENGELMORE@SUMEX-AIM)

Martin Epstein
National Institutes of Health
National Library of Medicine
Lister Hill Center
Bldg. 38A, Rm. 8S-810
Bethesda, MD 20209
(ARPANET: MEPSTEIN@NLM-MCS)

Steve Fickas
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291
(ARPANET: FICKAS@ISI)

Dr. Richard Fikes
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(ARPANET: FIKES@PARC-MAXC)

Professor Timothy W. Finin
Dept. of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
33rd and Walnut Streets
Philadelphia, PA 19104
(ARPANET: TIM.UPENN@UDEL)

Mark S. Fox
Department of Computer Science
Carnegie-Mellon University
Schenley Park
Pittsburgh, PA 15213
(ARPANET: MARK.FOX@CMUA)

Dr. Michael W. Freeman
R&D/FSSG
Burroughs Corporation
P.O. Box 517
Paoli, PA 19301
(ARPANET: FREEMM@WHARTON-10)

Alan Frisch
Department of Computer Science
Mathematical Sciences Building
University of Rochester
Rochester, NY 14627

Jeffrey A. Gibbons
Department of Artificial Intelligence
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: JGIBBONS@BBNG)

Randy Goebel
Department of Computer Science
University of British Columbia
2075 Wesbrook Mall
Vancouver, British Columbia V6T 1W5
Canada

Bradley A. Goodman
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: BGOODMAN@BBNG)

Dr. Norton R. Greenfeld
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: GREENFELD@BBNG)

Dr. Carole Hafner
Computer Science Department
General Motors Research Laboratories
Warren, MI 48090

Professor Patrick Hayes
Computer Science Department
Rochester University
Rochester, NY 14627

Dr. D. Austin Henderson
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(ARPANET: AHENDERSON@PARC-MAXC)

Dr. David J. Israel
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: DISRAEL@BBNG)

Professor Aravind Joshi
Dept. of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
33rd & Walnut Streets
Philadelphia, PA 19104
(ARPANET: JOSHI@WHARTON)

Tom Kehler
Knowledge-Based Systems Branch
Computer Science Library
Box 226015, MS238
Texas Instruments
Dallas, TX 75266
(ARPANET: TKEHLER@BBNA)

Bryan Kramer
Dept. of Computer Science
McLennan Labs
University of Toronto
Toronto, Ontario M5S 1A7
Canada

Robert Krovetz
National Institutes of Health
National Library of Medicine
Lister Hill Center
Bldg. 38A, Rm. 8S-810
Bethesda, MD 20209
(ARPANET: KROVETZ@NLM-MCS)

Professor Henry Leitner
Department of Computer Science
Boston College
Fulton Hall
Chestnut Hill, MA 02167
(ARPANET: LEITNER@HARV-10)

Dr. Hector Levesque
Artificial Intelligence Research Laboratory
Fairchild Research and Development
4001 Miranda Ave.
Palo Alto, CA 94304
(ARPANET: LEVESQUE@SRI-KL)

Dr. Robert W. Lingard
Lockheed California Company
P.O. Box 551
Burbank, CA 91520

Department 80-23
Plant A-1, Building 67

Thomas A. Lipkis
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291
(ARPANET: LIPKIS@ISIF)

Dr. William C. Mann
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291
(ARPANET: MANN@ISIB)

Dr. William S. Mark
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291
(ARPANET: MARK@ISIF)

Dr. Larry M. Masinter
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(ARPANET: MASINTER@PARC-MAXC)

Christian Mathiesson
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291

Eric Mays
Dept. of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
33rd & Walnut Streets
Philadelphia, PA 19104
(ARPANET: MAYS@WHARTON)

Professor David D. McDonald
Department of Computer and Information Science
Graduate Research Center
University of Massachusetts
Amherst, MA 01003
(ARPANET: DDM@MIT-AI)

Dr. Lorenzo Moretti
Instituto di Linguistica Computazionale
Via Della Faggiola 32
56100 Pisa
Italy

Cdr. Ron Ohlander
DARPA
1400 Wilson Blvd.
Arlington, VA 22209
(ARPANET: OHLANDER@ISI)

David Oppenheim
Decision Aiding Systems Laboratory
Department of Decision Sciences
The Wharton School
University of Pennsylvania
Philadelphia, PA 19104

Nicholas Palevsky
Aiken Computation Laboratory
Division of Applied Sciences
Harvard University
Cambridge, MA 02138
(ARPANET: NPALEVSKY@BBNG)

Martha Stone Palmer
Dept. of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
33rd and Walnut Streets
Philadelphia, PA 19104

Peter Patel-Schneider
Department of Computer Science
McLennan Labs
University of Toronto
Toronto, Ontario M5S 1A7
Canada

Professor C. Raymond Perrault
Department of Computer Science
McLennan Labs
University of Toronto
Toronto, Ontario M5 S1A7
Canada
(ARPANET: RPERRAULT@BBNG)

Dr. Rachel Reichman
Department of Electrical Engineering
 and Computer Science, C-015
University of California, San Diego
La Jolla, CA 92093
(ARPANET: REICHMAN@NPRDC)

Nathan Relles
Software Research
Sperry Univac, MS 2G3
Blue Bell, PA 19424

Professor Edwina L. Rissland
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003
(ARPANET: ERM@MIT-AI)

R. Bruce Roberts
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: BROBERTS@BBNG)

Ethan A. Scarl, A134
Mitre Corporation
Bedford, MA 01730
(ARPANET: SCARL@ECL)

James Schmolze
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(617)497-3617
(ARPANET: SCHMOLZE@BBNG)

Professor Stuart C. Shapiro
Department of Computer Science
SUNY at Buffalo
4226 Ridge Lea Road
Amherst, NY 14226

Dr. Candace L. Sidner
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: SIDNER@BBNG)

Dr. Brian C. Smith
Xerox Corporation
Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94304
(ARPANET: BrianSmith@PARC-MAXC)

Professor Elliot M. Soloway
Yale University
10 Hillhouse Road
New Haven, CT 06520

Dr. Norman K. Sondheimer
Mailstop 2G3
Software Research
Sperry Univac, Division Hdqtrs.
P.O. Box 500
Blue Bell, PA 19424
(ARPANET: SONDHEIMER@ECL)

William Swartout
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291

Dr. Harry Tennant
P.O. Box 225936
Mail Station 371
Dallas, TX 75265
(ARPANET: TENNANT@BBNA)

Fred Tou
(Graduate student at MIT)
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94301

Lynn Tranchell
Dept. of Computer Science
SUNY at Buffalo
4226 Ridge Lea Road
Amherst, NY 14226

John J. Vittal
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: VITTAL@BBNG)

Professor David L. Waltz
Coordinated Science Laboratory
University of Illinois
Urbana, IL 61801

Professor Bonnie L. Webber
Dept. of Computer and Information Science
The Moore School of Electrical Engineering
University of Pennsylvania
33rd and Walnut Streets
Philadelphia, PA 19104
(ARPANET: WEBBER@BBNG)

E. Judith Weiner
Computer and Information Sciences
Computer Activity Building
Temple University
Philadelphia, PA 19122

Professor Ralph Weischedel
Department of Computer and Information Sciences
University of Delaware
Newark, DE 19711
(ARPANET: WEISCHEDEL@UDEL)

Dr. David Wilczynski
USC/Information Sciences Institute
4676 Admiralty Way
Marina del Rey, CA 90291
(ARPANET: WILCZYNSKI@ISIF)

Mike Williams
Xerox Palo Alto Research Center
3333 Coyote Hill Road
Palo Alto, CA 94301

Gerald Wilson
Computer Corporation of America
575 Technology Square
Cambridge, MA 02139
(ARPANET: WILSON@CCA)

Beverly Woolf
Department of Computer and Information Science
University of Massachusetts
Amherst, MA 01003

Dr. William A. Woods
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: WOODS@BBNG)

Dr. Martin D. Yonke
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: YONKE@BBNG)

Frank Zdybel
Bolt Beranek and Newman Inc.
50 Moulton Street
Cambridge, MA 02238
(ARPANET: ZDYBEL@BBNG)

APPENDIX D
SUMMARY OF THE KL-ONE LANGUAGE

In this appendix, we present an overview of the KL-ONE language. This discussion should provide a conceptual summary of the types of KL-ONE objects and their structural relations to one another.

D.1 Language Structure and Philosophy

KL-ONE is a language for the explicit representation of conceptual information based on the idea of structured inheritance networks [Brachman 78, Brachman 79]. Before going into the details of KL-ONE structures, we will first paint the background of our philosophy in developing the language.

As mentioned, KL-ONE is intended to represent general conceptual information by allowing the construction of a knowledge base of a single reasoning entity (although it has been used as a repository for information from multiple sources). A KL-ONE network thus represents the beliefs about the world (and other possible worlds) as conceived by the system using it. Note that we are not intending to attempt to capture the world "as it really is" - only the conception of it by an individual perceiver.

KL-ONE can be thought of as having two sublanguages - a description language and an assertion language. The description language allows one to form a variety of description terms (e.g., general terms, individual descriptions) out of other description terms using a small set of primitive description-formation operators. This gives us an extensible repertoire of conceptual terms (i.e., a conceptual vocabulary) that can then be used to make assertions. For example, we would use only description-structuring in forming a compound KL-ONE description corresponding to "a man from Mars", using the KL-ONE descriptions for "a man" and "Mars". Simply forming this description asserts nothing about any particular man or planet since structures in the description language have no assertional import by themselves (but see Section D.3).

The assertion language, on the other hand, makes use of terms from the description language to make statements about the world. The assertion language is somewhat impoverished as compared, say, to a first order language with equality, but it includes statements of coreference of description (in a particular context), and of existence and identity of individuals (in a particular context). For example, we might form a description like "the person giving the talk", and use it to assert that such a person exists. We could then establish another description - say, "a man from Mars" - as coreferential with the first, and thereby make the statement, "the person giving the talk is a man from Mars". This latter type of construct is a legitimate object of belief (whereas descriptions are not).

In the past, most of our work on KL-ONE has been on description-formation; until recently very little attention was paid to making assertions (it is felt that existing paradigms -- e.g. predicate logic -- are adequate for this task). While we have now begun to focus more intensively on the assertion language, the system that is described herein handles description structure well and has only a crude mechanism for making assertions.

D.1.1 Epistemological Primitives

KL-ONE is, in a sense, an "object-centered" language. Its development has proceeded from traditional semantic networks, but its principal structures are not based on either propositions or sets as were those of several earlier semantic net systems (e.g., see [Schubert, et al. 79] and [Hendrix 79]). Instead, the principal element of KL-ONE is the structured conceptual object, or Concept.

Our view of these objects comes from a careful analysis of early trends in semantic networks and more recent trends in knowledge representation in general. As discussed in [Brachman 79] and [Woods 75], the history of network representations is fraught with imprecision on the meanings of nodes and links. We have found links in networks to represent implementational pointers, logical relations, semantic relations (e.g., "cases") and arbitrary conceptual and linguistic relations. Network schemes consistent with structures of any one of these "levels" (implementational, logical, conceptual, linguistic - see [Brachman 79]) can be compared and tested for adequacy, but unfortunately, most of the existing paradigms mix structures from two or more of these levels. This makes for confusing notation and difficulty in explaining a system's interpreter.

Realizing the value of consistency at a single level of network primitive, we have set out to capture an adequate set of primitive elements for structuring a broad spectrum of kinds of concepts. We are attempting to determine a reasonable set of underlying object and relation types for generalized knowledge-structuring. To the extent that we can formalize the language of concepts in a grammar for well-formed conceptual structures, we have defined what might be called an "epistemology". This is not a theory of any particular domain - one builds that on top of this level - but a generative theory of the structure and limits of thought in general. We have addressed our work on KL-ONE to an epistemological level of network primitives with the long-term goal of examining the scope of what is "thinkable".

KL-ONE thus comprises a set of "epistemologically primitive" structure types and structure-forming operations. We have attempted to understand the important features of the internal structure of concepts, and to embody them in a language that is expressively powerful and fairly natural to use.

D.2 Concepts and Roles

As mentioned, the principal elements of KL-ONE descriptions are Concepts, of which there are two major types - Generic and Individual.[2] A Generic Concept is expected to act like the conceptual equivalent of a "general term" [Quine 60] - potentially many individuals in any possible world can be described by it. Individual Concepts are similar but can describe one individual at most (in a particular context). A simple (albeit, incomplete) view of the difference between Generic and Individual Concepts is captured in the following contrast: "a man from Mars" (Generic) versus "the man from Mars" (Individual).

Generic Concepts are arranged in a definitional taxonomy which represents their subsumption relationships. We refer to this taxonomy as definitional because the meaning of any particular Concept is derived, at least in part, from its location in the taxonomy. For example, by having the Concept HUMAN[3] subsume the Concept MAN, one simultaneously defines part of the latter, in that all components of the definition of HUMAN apply to MAN. The KL-ONE system provides inheritance facilities to support this property of its taxonomies. It is important to note, however, that the Concept HUMAN does not derive any of its meaning from the Concept MAN. In general, a Concept's meaning is strictly determined by the definitions of its subsuming Concepts, plus information located specifically with the Concept (the manner of this

[2]There is a third type of Concept in KL-ONE - the ParaIndividual - which we will get to in Section D.5.

[3]From this point on, we will use "Concept" to mean "Generic Concept", except when "Concept" alone would be ambiguous.

"local" information, as well as the inheritance facilities, will be described throughout the remainder of this paper).[4]

KL-ONE taxonomies always have a single root Concept (which currently is named THING) of which all other Concepts are descendants. In Figure 1 we present a simple taxonomy. Each ellipse represents a Generic Concept; each link between Generic Concepts (the wide arrow) represents a subsumption relationship where the "higher" Concept (the one being pointed to) subsumes the Concept "below" it (the one being pointed from). Subsumption is a transitive relation, so a Generic Concept actually subsumes all Generic Concepts "below" it.[5] In KL-ONE terminology, the subsumption relation is indicated by a superC cable (the wide arrow) and the subsumer and subsumee Generic Concepts, respectively, are called the superConcept and subConcept.

You will note in Figure 1 that a Generic Concept can have many superConcepts and/or subConcepts. The Concept MAN has both MALE-ANIMAL and HUMAN as its superConcepts. Hence, both MALE-ANIMAL and HUMAN subsume MAN and the Concept MAN is defined as the conjunction of the two. The use of multiple subConcepts (e.g., for the Concept ANIMAL) allows one to distinguish the various subtypes or subclasses of a Concept (four types of ANIMAL are included in Figure 1).

We have stated that Concepts inherit at least part of their definition from their superConcepts. In fact, KL-ONE places a strong emphasis upon precise definitions for Concepts, where that definition is expressed explicitly in the KL-ONE network. This emphasis has several important effects regarding the design of KL-ONE. In particular, names for KL-ONE objects (such as Concepts) are not used by the system in any way. They are merely conveniences for the user. Also, "cancellation" of parts of a definition that a Concept inherits from its superConcepts is explicitly prohibited (this is addressed more completely in Section D.3.3).

The components of a KL-ONE Concept which constitute its entire definition are:

---

[4]There has been considerable discussion among KL-ONE users as to whether or not KL-ONE should be able to represent exhaustion or mutual exclusion among a Concept's subsumees. If it did, then a Concept could gather part of its definition from those it subsumes, in the case of exhaustion, or "sibling" Concepts (Concepts which share the same subsuming Concept), in the case of mutual exclusion. Currently, however, KL-ONE does not support either.

[5]From this point on, the notions of "above" and "below" will be used interchangeably with "subsumer" and "subsumee", respectively.

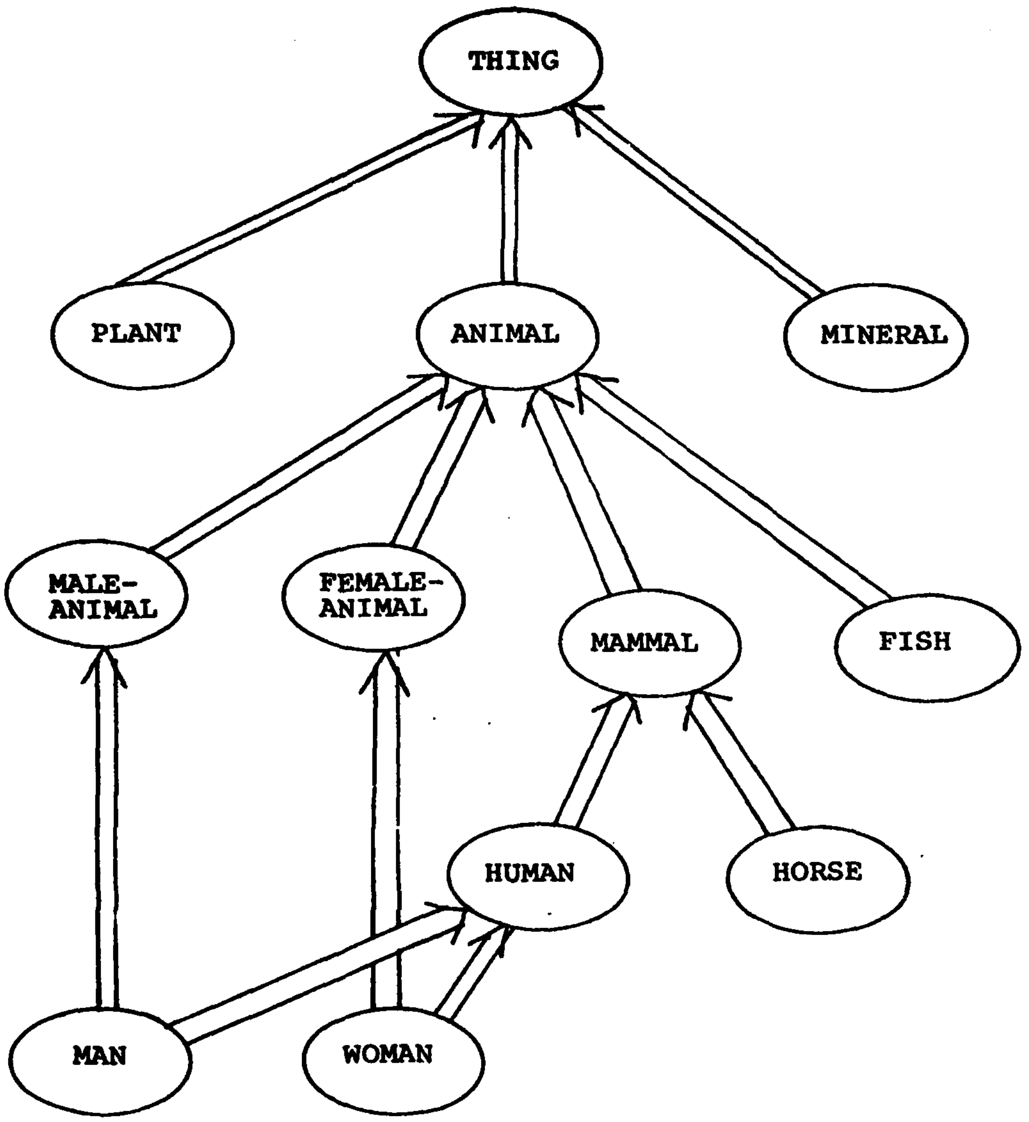


FIG. 1. A SIMPLE KL-ONE NETWORK OF GENERIC CONCEPTS

- o the definitions of its subsuming Concepts (its superConcepts),
- o its local internal structure expressed in
  - . Roles, which describe potential relationships between instances of the Concept and other closely associated Concepts (i.e., those of its properties, parts, etc.), and
  - . RoleSet Relations, which express the interrelations among the functional Roles.

A superConcept serves as proximate genus, while the internal structure expresses essential differences, as in classical classificatory definition [Sellars 17].[6] The summum genus is represented by the Concept THING. THING is the only Concept that has no superConcepts.

A pictorial representation of THING is shown in Figure 2. The Concept itself is represented by the ellipse labeled with its name; the Role subpart is represented by the encircled square. As is evident here, Roles themselves have structure, including descriptions of potential fillers (called "Value Restrictions", or V/R's), and names. In Figure 2 we see that the Role named subpart has a Value Restriction of THING. This says that potential subparts of THINGs must themselves be THINGs. While here such a restriction is vacuous (there is no other possibility), the system interprets Value Restrictions as necessary type restrictions on Role fillers. No "cancellation" of such restrictions is allowed. Cases arise where several Value Restrictions are applicable to a Role filler (these cases will be apparent as the inheritance mechanism is explained). If more than one V/R is applicable at a given Role, the restrictions are taken conjunctively.

There are two basic kinds of Roles in KL-ONE: RoleSets andIRoles (for 'Instance Roles'). RoleSets capture the notion that a given functional role of a Concept (e.g., upright of an arch, officer of a company, input to a program) can be filled in the same instance by several different entities. On a Generic Concept, a RoleSet captures the commonality among a set of individual role players (e.g., what all officers a given company will have in

[6]The intuitive form of this type of definition will be evident from the "JARGON" statements in figures to follow. JARGON is a highly specialized, English-like language for describing KL-ONE objects and relationships. It has two important properties: it is usually easy to understand the KL-ONE structures being described by a JARGON statement, and an interpreter exists which can translate most JARGON statements into appropriate KL-ONE structures.

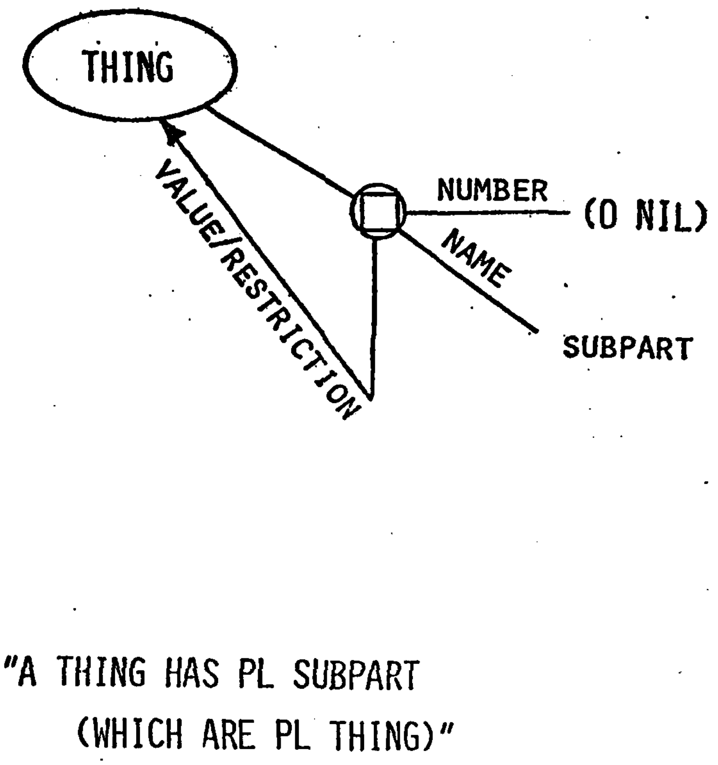


FIG. 2. THE MOST GENERAL CONCEPT, THING.

common). Any object described by the Concept will have its own set of role players, so a version of the Concept that describes a particular individual - an Individual Concept - will have a corresponding set of role descriptions, each describing the binding of a given filler into the functional role (e.g., "the first-officer of the Enterprise is the man from Vulcan"). IRoles are the KL-ONE structures representing these individual bindings (see below).

Since the functional roles defined by RoleSets can be played by more than one individual at a time, RoleSets have Number Restrictions to express cardinality information. At the moment, a Number Restriction is a pair of numbers (or NIL) defining the range of cardinalities for sets of role-player descriptions (NIL means "don't care"). Thus the Number Restriction in Figure 2 indicates that any THING can have arbitrarily many subparts. All other facets of the Generic RoleSet description are applicable individually to each subpart (e.g., each will have the functional role name subpart attributable to it, and each subpart must satisfy the Value Restriction).

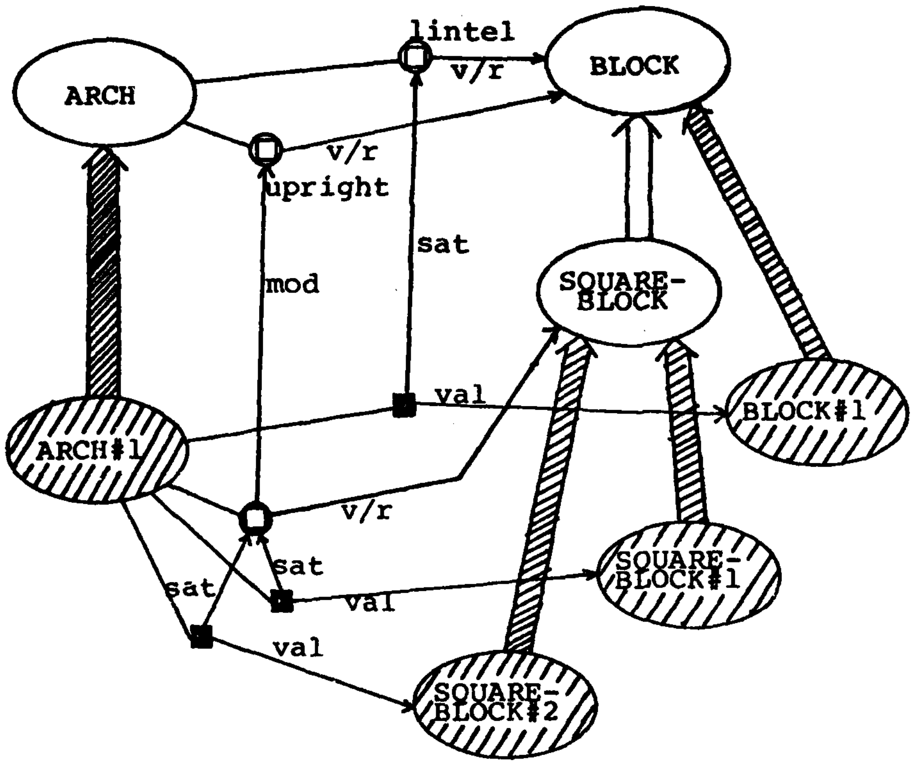


FIG. 3. IROLES AND PARTICULAR ROLESETS

A RoleSet on an Individual Concept stands for the set of role descriptions for that Concept only (e.g., the particular set of uprights of a particular arch) - it is not a generic description. In such a case, we call the Role a "Particular RoleSet". IRoles appear only on Individual Concepts, and are used to represent particular bindings of Roles to Individual Concepts (e.g., the lintel of a particular arch is a particular block). Since a RoleSet describes a set of fillers, when there are multiple fillers for a single RoleSet, several IRoles must be used - one IRole for each binding. Figure 3 shows a Generic Concept named ARCH that has two RoleSets, upright and lintel. An Individual Concept of an ARCH, named ARCH#1, is below it (Individual Concepts are depicted as shaded ellipses). ARCH#1 is called an individuator of ARCH - all Individual Concepts must individuate some Generic Concept. The shaded arrow connecting the two is called an Individuation Cable. ARCH#1 has a Particular RoleSet for upright (drawn as a RoleSet with the circle filled in) that states that all fillers for ARCH#1's uprights are not only BLOCKs (the Value Restriction inherited from the ARCH Generic Concept), they are also SQUARE-BLOCKs. The IRoles (filled-in squares) show the particular BLOCKs which constitute ARCH#1.

Figure 4 motivates the distinction between Roles and their fillers. Contrast the intended referent of the word "it" in sentences a and b:

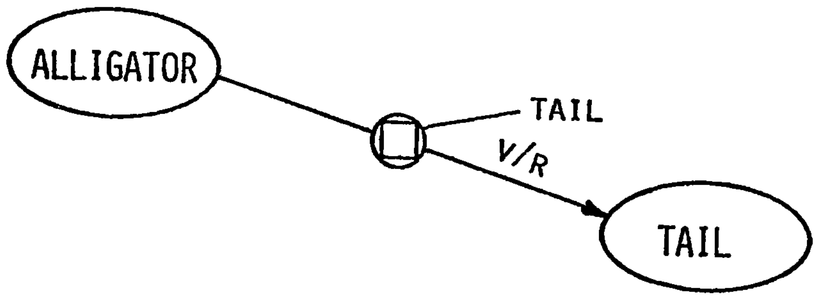


FIG. 4. A ROLE IS NOT THE SAME AS A ROLE-PLAYER.

- o The alligator's tail fell off.
  - a. It lay wriggling on the ground.
  - b. It will grow back again.

The "tails" that the pronouns are intended to refer to are different in the two cases. In the first, the Role name is being used to refer to the previous filler. In the second, since it is not the previous tail that will grow back, the name must be referring to something more abstract. We take the intent of this latter reference to be represented by a KL-ONE Role.

D.3 Derivative Definition

As mentioned above, definitional generic knowledge in KL-ONE is expressed in a taxonomic inheritance structure. The backbone of such a network is formed by inter-Concept inheritance Cables[7] which pass the structure of definitions. The inheritance Cable is the primary description-formation operator of KL-ONE. It specifies that the meaning of a Concept is to be determined by conjoining the meanings of its superConcepts.

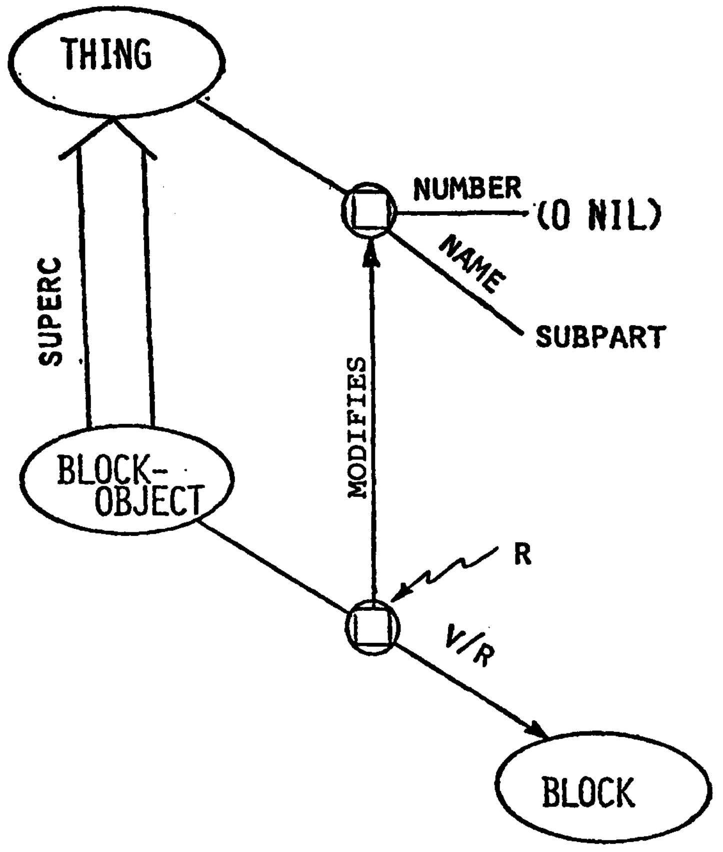
Note that we have not said that the Cable merely asserts the subclass relation between two independently defined classes. The subConcept is defined in terms of the superConcept. This definitional relationship is akin to lambda abstraction, such as that between A and B in B(x) = lambda(x)[A(x) & ...], and is the basis for our ability to build an analytic classifier into the KL-ONE interpreter (see Section D.3.1).

Figure 5 illustrates how to define a simple Concept from one we have already defined. The Cable is depicted as a double-shafted arrow. The lower Concept constructs a more specific meaning from the meaning of THING by restricting the subpart Role - in this case, to have V/R BLOCK (which is, for now, an undefined type of THING). The Modifies link[8] from the Role indicated "r" to the subpart role of THING indicates that the fillers of the subpart Role of the lower Concept are restricted to be BLOCKs. Role r is the subpart Role for the lower Concept, and thus inherits all of its meaning from that superRole. The new V/R is taken conjunctively with the old one (to repeat, you can not cancel an inherited V/R), and the Name and Number Restriction are inherited intact. As a result, the lower Concept is "a thing whose subparts are blocks". All other Roles from the upper Concept that are not modified by the lower Concept are inherited intact by the lower one (of course, this example does not demonstrate this property since THING has only one Role).

In Figure 5, we call this lower Concept BLOCK-OBJECT. BLOCK-OBJECT is not intended to be some observationally determined class, but merely a _new term in the description language_, defined to be nothing more than "A THING

[7]The superConcept link, which was introduced earlier, is only one type of inheritance Cable.

[8]The link is not independent of the SuperC Cable. It can be thought of as part of the Cable.



"A BLOCK-OBJECT IS A THING WHOSE PL SUBPART ARE PL BLOCK"

FIG. 5. BASIC DESCRIPTION FORMATION.

WHOSE PL[9] SUBPART ARE PL BLOCK". The SuperC Cable between BLOCK-OBJECT and THING serves only as a use of the latter in the definition of the former. Because BLOCK-OBJECT is defined in terms of THING, it must of course be true that (in any possible world) any BLOCK-OBJECT is a THING.

---

[9]PL is the JARGON morpheme for expressing the plural form of a noun. Read "SUBPARTS" for "PL SUBPART" and "BLOCKS" for "PL BLOCK".

D.3.1 Taxonomy and the Classification Algorithm

Since "BLOCK-OBJECT" means "A THING WHOSE PL SUBPART ARE PL BLOCK", any new term derived from BLOCK-OBJECT will of necessity carry "THING" as part of its meaning. By the same token, any entity with "a thing whose subparts are blocks" as part of its description will necessarily be a BLOCK-OBJECT. KL-ONE enforces this subsumption of Concepts by guaranteeing that a Concept entered in the network will be placed below all other Concepts that definitionally subsume it and above all Concepts that it subsumes. This process of finding the correct "location" for a Concept is called classification and it is one of the features of KL-ONE that makes it unique among current representation languages - the interpreter in a sense "understands" the Concept-formation language, and keeps all Concepts in a strict subsumption taxonomy based on their internal structures.

Thomas Lipkis, of USC/ISI, has written a program that performs the classification just described. The algorithm is complete with respect to the knowledge-structuring primitives that are described in this paper. It is described in Lipkis' paper (see page 128), where he also discusses the classifier's inferential limitations. In particular, he explains why one cannot expect the classifier to "understand" knowledge that is extrinsic to KL-ONE (e.g., number theory).

D.3.2 Incompletely Defined Terms

Many uses of KL-ONE require the introduction of terms that cannot be completely defined via the term-forming operators introduced so far.[10] To meet this need, KL-ONE provides a mechanism for introducing primitive terms from which other terms may be defined - leaving the meaning of these primitive terms to processes outside KL-ONE. Such primitive terms are represented as Concepts whose definition, viz-a-viz KL-ONE, is incomplete. They are called "starred Concepts", and are depicted with a "*" alongside the ellipse representing the Concept (starred Concepts have also been called "magic" Concepts).

The use of starred Concepts lets the KL-ONE system distinguish between terms which are primitive from terms whose definition is completely within the

[10]In particular, some domains require "natural kind" terms (e.g., elephant), the meaning of which, many researchers claim, cannot be fully analyzed.

taxonomy. This distinction is critical to the correct operation of the classifier, which must prevent a Concept from becoming a specialization of a starred Concept, unless that particular subsumption relation is stated explicitly. For example, let ELEPHANT be a starred Concept with various Roles. If a second description matches that of an ELEPHANT, it need not be an ELEPHANT - the structure of ELEPHANT constitutes necessary, but not sufficient, conditions for recognizing an ELEPHANT. The classifier would never infer that the second description was an ELEPHANT unless explicitly stated. Concepts that are not starred are interpreted differently - their structure provides the conditions that are both necessary and sufficient for recognition.

Normally, these primitive terms are used as part of the definition of more complex Concepts. For example, we can introduce a starred Concept for POLYGON that has a RoleSet named side, whose Number Restriction is from 3 to infinity. A TRIANGLE can then be defined completely; it is a POLYGON whose side's number restriction is exactly 3. If a new Concept is classified that is a POLYGON with exactly 3 sides plus other structure, it will become a specialization of TRIANGLE. There are two crucial ingredients to this inference: the new Concept was defined to already be a POLYGON (the classifier would never make that inference), and it satisfied the conditions of TRIANGLE viz-a-viz its KL-ONE structure.

D.3.3 A note on "cancellation"

The lack of cancellation of Value Restrictions might appear problematic from the point of view of representing "exceptions" (e.g., three-legged elephants - see [Fahlman 79]). However, if we allow cancellation of properties within Concepts, then these properties are reduced in status from definition to default assertions. The KL-ONE taxonomy <u>defines</u> terms; non-definitional information <u>about</u> terms, such as defaults assertions, is more appropriately expressed elsewhere. Furthermore, cancellation would derail the classifier.[11] We intend, instead, to allow statements of default rules <u>between Concepts</u> only. Thus (when implemented), one would not represent elephants' typically having four legs as in Figure 6. Instead, one would assert something like

[11]The classifier has its hands tied if Roles express defaults: a given Concept can be forced to fit almost anywhere, since all we have to do is cancel the Roles that don't match up. In this way, THREE-LEGGED-ELEPHANT can just as well be above FOUR-LEGGED-ELEPHANT as below it. See [Brachman 80] for more about this problem.

$$\frac{\text{Elephant}(x) : M[\text{Four-legged-mammal}(x)]}{\text{Four-legged-mammal}(x)}$$

in the manner of [Reiter 80]. That is, "unless you have information to the contrary, assume of an elephant that it is also a four-legged-mammal". This leaves the Concepts of ELEPHANT and FOUR-LEGGED-MAMMAL distinct (as they should be), and inviolate. KL-ONE seems to be different from many of today's representation languages precisely because it uses strictly complex predicates, such as "four-legged-mammal."

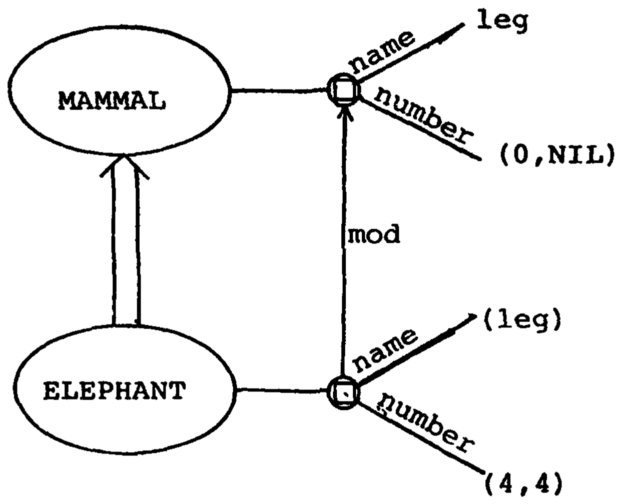


FIG. 6. NOT THE WAY TO DESCRIBE ELEPHANTS.

D.4 Inter-Role Relations

The "Modifies" link of Figure 5 is only one of four types of KL-ONE links for expressing inter-Role inheritance relationships. Such relationships bear the brunt of the "structured inheritance" carried by SuperC Cables.

The four types of relationship are summarized here:

- o restriction (of filler description and/or number)[12]; e.g., given that a COMPANY has officers, each of whom is a PERSON, a particular kind of COMPANY will have exactly three officers, all of whom must be over 45.
- o differentiation (of a Role into subRoles); e.g., differentiating the officers of a COMPANY into president, vice-president, etc. This is a relationship between RoleSets in which the more specific Roles inherit all properties of the parent Role except for the Number Restriction (since that applies to the set and not the fillers);
- o particularization (of a RoleSet for an Individual Concept); e.g., the officers of BBN are all COLLEGE-GRADUATEs; this is the relationship between a RoleSet of an Individual Concept and a RoleSet of a parent Generic Concept.
- o satisfaction; this is the relationship between an IRole and its parent RoleSet.

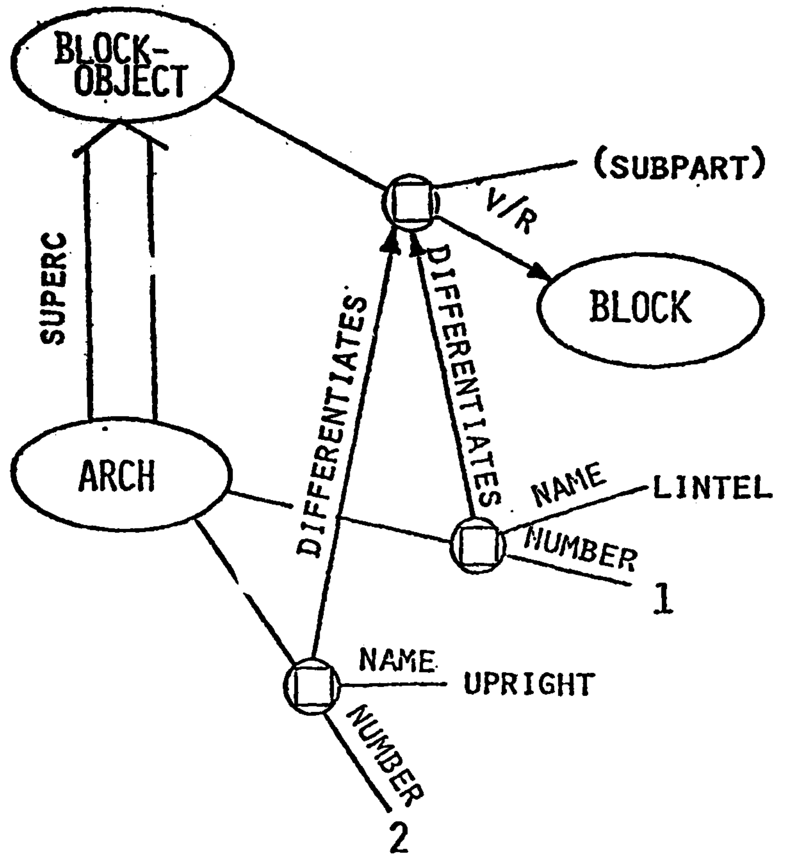
Role differentiation is one of KL-ONE's unique features; Figure 7 illustrates its use. Since RoleSets have an associated cardinality, they can be divided into "sub-Role Sets". In the figure, we define the Concept of an ARCH as a BLOCK-OBJECT, one of whose subparts is its lintel, and two of whose subparts are its uprights. Since each of these Roles is also describable in the more general terms of its superRole, they each inherit all of the structure of that Role. Typically, the "lower" Role will have a more highly constrained Number Restriction than the "higher" Role, as in the case of our lintel and upright. However, the Number Restriction of the "lower" Role is the conjunction of its own Number Restriction and that of the "higher" Role.

The purpose of RoleSet differentiation is to allow further specification of subsets of RoleSets - the RoleSet can have sub-types, each of which is explicitly described. This is akin to the relationship between a Concept and its subConcepts, and is supported by full inheritance from the "higher" RoleSet to its "lower" RoleSets. We intend to generalize the differentiation mechanism to allow multiple partitions of a RoleSet - at the moment we allow only one.

It should be noted that differentiation of a Role can, and should, occur within a single Concept. RoleSets are inherited through SuperC Cables, so an equivalent alternative to the previous definition of ARCH is that expressed in

---

[12]We intend at some point to change the name of the "Modifies" link to "Restricts" to reflect this.



"AN ARCH IS A BLOCK-OBJECT WHICH HAS A LINTEL AND 2 UPRIGHT"

"THE LINTEL OF AN ARCH IS ONE OF ITS PL SUBPART (AS A BLOCK-OBJECT)"

"THE UPRIGHT OF AN ARCH ARE SOME OF ITS PL SUBPART"

FIG. 7. DIFFERENTIATION.

Figure 8. The Modifies link expresses the fact that the RoleSet seemingly

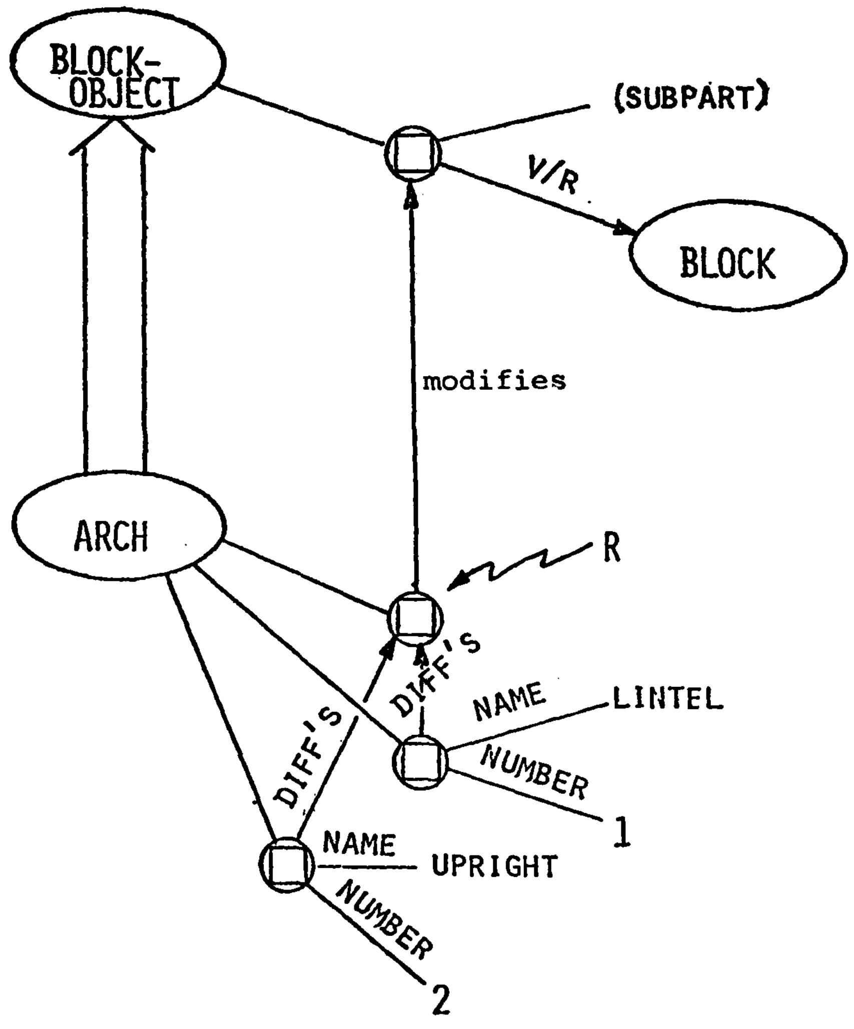


FIG. 8. INTERNAL DIFFERENTIATION.

local to ARCH is the very same one as its superRole. Here there are no further restrictions on ARCH's subpart, so this is simply making explicit a structure that was for all intents and purposes already there. As far as

KL-ONE is concerned, Role R is "as good as there" in Figure 7.[13]

Also, we note that subRoles can themselves be modified (by Concepts defined in terms of the one where the subRole appears) or differentiated (as long as their cardinality is greater than one).

D.5 RoleSet Relations

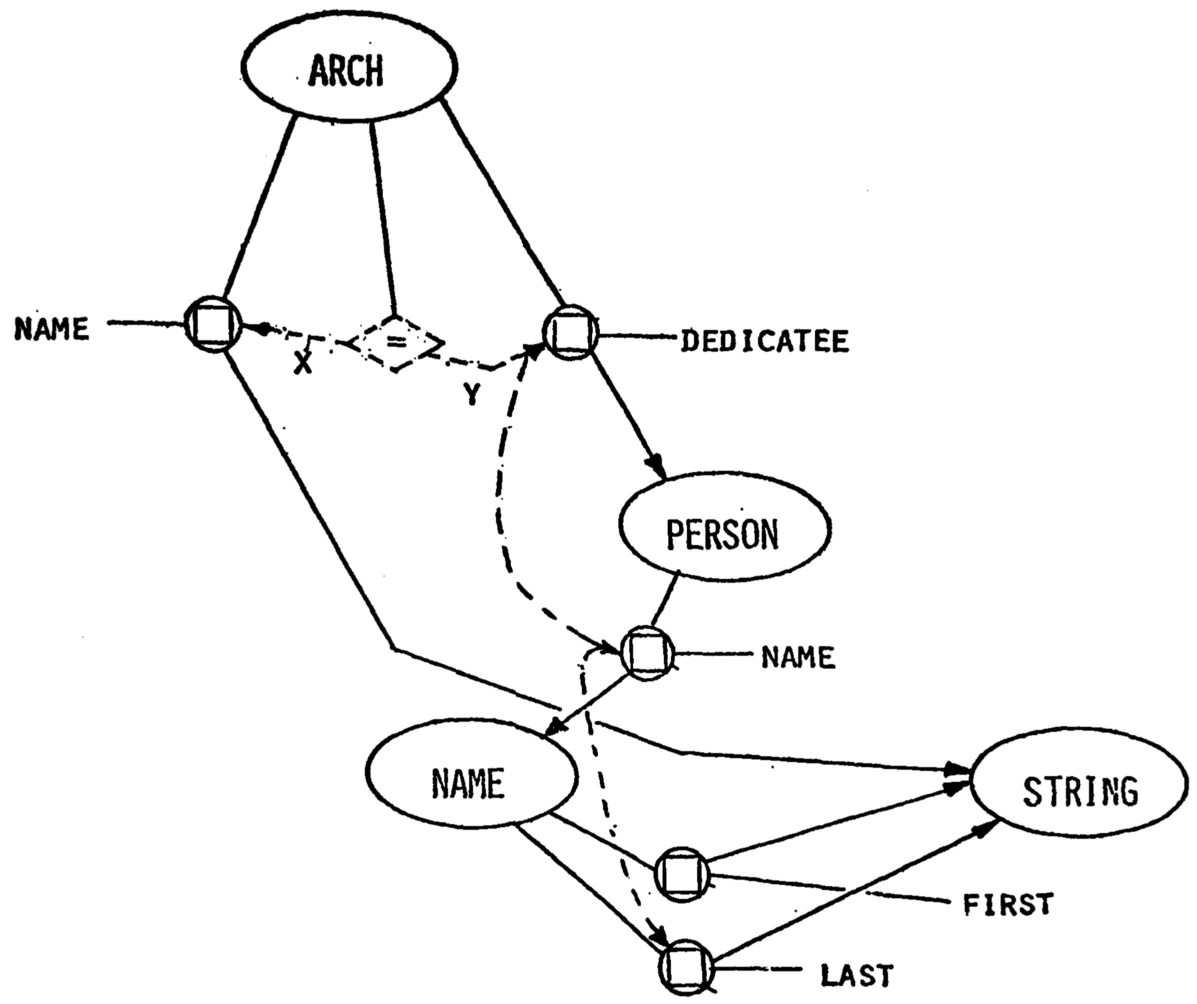
While KL-ONE Roles are assigned local "names", and inherit them from superRoles as well, these are meaningless strings as far as the system is concerned. Thus, in the structure so far described, we have seen how Roles describe actual or potential Role-fillers, but nothing (except our wishful thinking) gives the Role its intended meaning as a functional role description. In order to provide for the explicit representation of the roles that Role-fillers play, plus provide for the representation of inter-Role relations, we complete the structure of a KL-ONE Concept with a set of RoleSet Relations (RSR's).

The first type of RSR we will examine is the RoleValueMap (RVM). An RVM expresses a simple relationship between two sets of Role-fillers - either identitv or inclusion. An RVM can equate the sets of fillers (N.B. not IRoles) of two Roles of the same Concept, or a Role of an Individual Concept with a Role of another Concept. Figure 9 illustrates the RVM structure. Imagine that we have augmented our ARCH description with the following:

1. AN ARCH HAS A NAME WHICH IS A STRING
2. AN ARCH MUST HAVE A DEDICATEE WHICH IS A PERSON
3. A PERSON MUST HAVE A NAME WHICH IS A NAME
4. A NAME MUST HAVE A FIRST WHICH IS A STRING AND HAVE A LAST WHICH IS A STRING.

These JARGON sentences define the non-dashed structure in the figure.

[13]Its "being there" or not is an artifact of the implementation. Regardless of the implementation, the Role is there as far as KL-ONE is concerned, be it explicit or implicit.



"THE NAME OF AN ARCH IS THE SAME AS THE LAST OF THE NAME OF ITS DEDICATEE"

FIG. 9. A ROLEVALUEMAP.

Now, suppose we wanted to give some further indication of the meaning of an

ARCH's name; in particular, that it is the same as the last name of the person for whom the arch is dedicated. The dashed structure in Figure 9 shows the appropriate RoleValueMap (the diamond). The RVM has two pointers: x, to the Role name of ARCH, indicating "THE NAME OF AN ARCH"; and y, a role chain, indicating "THE LAST OF THE NAME OF THE DEDICATEE OF AN ARCH" (the first pointer, x, is also a role chain, albeit a short one). The RVM is hung off of ARCH, since it is part of the definition of ARCH (and not of PERSON, NAME or STRING). Note that if any of the Roles in the y chain had potentially multiple fillers, that chain would "evaluate", in an instance, to the complete set of STRINGs obtained by iterating over all dedicatees and all of their names, and all of their lasts.

Because the RoleValueMap in Figure 9 is a part of ARCH's definition, each instance of ARCH satisfies the generic relationship defined therein. That is, the set of STRINGs obtained by retrieving the names of a particular ARCH is the same as the set retrieved as the lasts of the names of the dedicatees of the very same arch.

One final note on the RVM is needed to motivate the use of a chained pointer. If the RVM were to point directly to the ultimate Role in the y chain of Figure 9, that pointer to the last Role of NAME would happen not to be problematic. However, if we had chosen to make the V/R of the name Role of ARCH be NAME as well, then the direct pointer would fail to disambiguate between the last of the name of the ARCH itself and the last of the name of its dedicatee. Thus, the chained pointer that starts at a Role of the enclosing Concept is necessary. Reading from the RVM out, the y pointer might be read as the "DEDICATEE'S NAME'S LAST", illustrating the prominent position of the dedicatee Role. In fact, one can think of role chains as a variation of functional notation, e.g., "last(name(dedicatee(ARCH)))".

The second type of RSR we will examine is the Structural Description (SD). SD's express how the Roles of the Concept interrelate and how they relate to the Concept as a whole via the use of parameterized versions ("ParaIndividual Concepts") of other Concepts in the network. Taking our ARCH example once again, we re-introduce the two RoleSets defined for Figure 7, lintel and upright. We can express the functional relationship between the lintel and uprights of an ARCH, namely that the uprights SUPPORT the lintel, with an SD that uses a ParaIndividual Concept. This is depicted in Figure 10 (for simplicity, we have not drawn the remainder of the ARCH Concept). The diamond in the figure depicts the SD. It contains a ParaIndividual Concept, SUPPORT#1 (double ellipse), which is a parameterized version of the SUPPORT Generic Concept. The arrow connecting SUPPORT and SUPPORT#1 is called a "ParaIndividuates Cable". The Roles on SUPPORT#1, called CorefRoles, have a one-to-one correspondence to the Roles on SUPPORT, indicated by the CorefSatisfies wires, and express their correspondence to the Roles on ARCH via Coref wires (Coref wires have also been called CorefValue wires). The intent of this SD is that as part of the definition of an ARCH, there must be

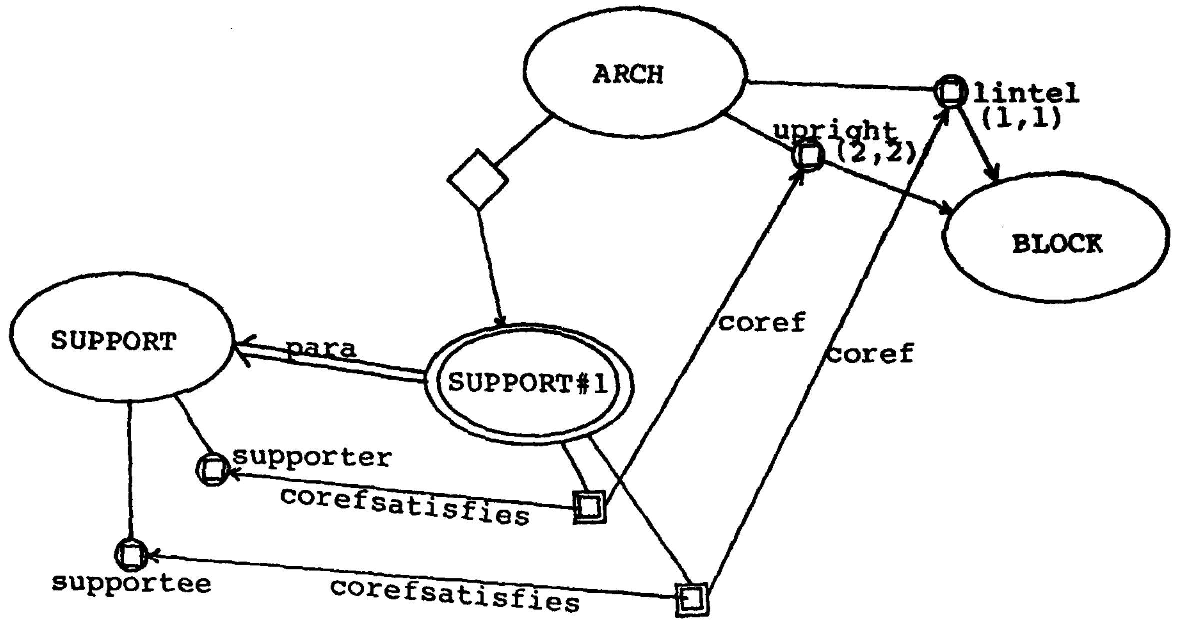


FIG. 10. A STRUCTURAL DESCRIPTION USING A PARAINDIVIDUAL CONCEPT

a lintel and more than one upright, where each upright must play the Role of supporter of some instance of SUPPORT in which the lintel (of the same ARCH) is the supportee.

The use of ParaIndividual Concepts has a natural analogy in programming languages. If one thinks of the Generic Concept of SUPPORT as being analogous to definition of some function F with arguments A and B, the ParaIndividual is akin to a call to F from some other function where the arguments, A and B, are substituted by other expressions. Programming languages typically rely upon argument order to make the correspondence between defined arguments and their use, whereas KL-ONE uses the wires mentioned above instead.

In general, RoleSet Relations express quantified relationships among RoleSets as set mappings. SD's are the subclass of RSR's that map over the Roles of a single Concept (the Coref wires must go through RoleSets of the Concept containing the SD). RSR's are intended to describe mappings among IRoles. They do this by specifying the mappings at the Generic Concept level. They are inherited through Cables and are restricted and particularized in a manner similar to that of Roles (for further discussion of RSRs, see the summary of the technical discussion on RSRs, page 44, and Freeman's paper on "Towards a calculus of structural descriptions ... ", page 115).

We should add one more note. In Figure 10, the Coref wires went directly from a CorefRole to a RoleSet. In general, Coref wires are actually role chains and have the same properties as role chains for RVMs (which was discussed earlier in this section). When used with ParaIndividual Concepts, the role chain can also point directly to the enclosing Concept in order to express the participation of the instance's "self" - that is, the thing as a whole - in a relationship. For example, in Figure 11, the Concept of a HUSBAND would have as part of its definition a RoleSet Relation describing a MARRIAGE in which the male-spouse Role was to be filled by the HUSBAND itself. For each instance of HUSBAND, there would have to exist a MARRIAGE description whose male-spouse was that instance of HUSBAND (and not some other HUSBAND).

D.6 Contexts and Nexuses

As mentioned earlier, the description formation part of KL-ONE has a complementary assertion-making part. We have tried carefully to distinguish between purely descriptional structure and assertions about coreference, existence, etc. All of the structure mentioned above (Concepts, Roles, and Cables) is definitional. All assertions are made relative to a Context and thus do not affect the (descriptive) taxonomy of generic knowledge. We anticipate that Contexts will be of use in reasoning about hypotheticals, beliefs, and wants.

One asserts the existence of some thing satisfying a description (i.e., Concept) by connecting it to a Nexus within a particular Context. This connecting link is called a Description Wire. A Nexus is a structureless entity which serves as a locus of coreference statements; it holds together various descriptions, all of which are taken to specify the same object in the Context. Nexuses have been conveniently thought of as corresponding to things in the world; KL-ONE, however, makes no such commitment. The Description Wires are also taken to be in the Context. Contexts are at the moment simply collections of Nexuses and Description Wires. Thus, a Context can act as a "world", which comprises a set of statements about existence and description

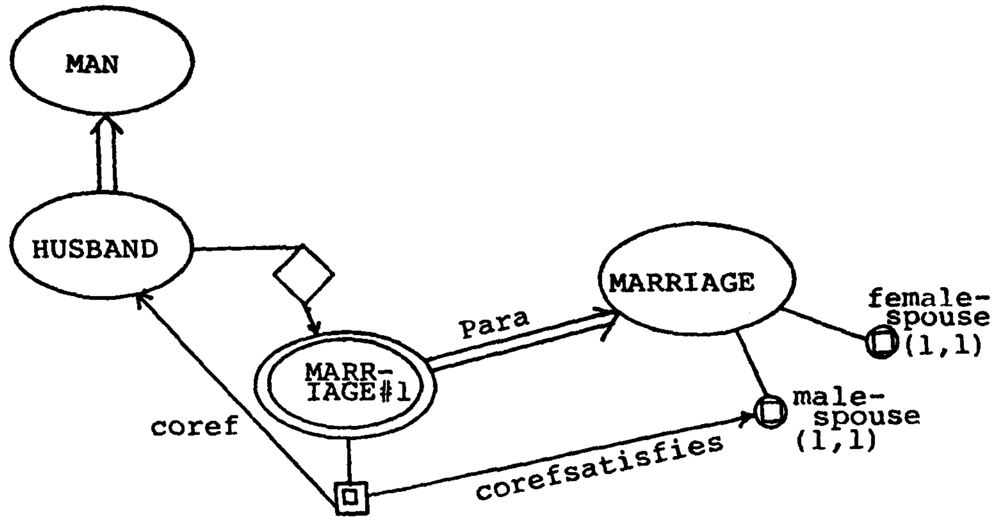


FIG. 11. A COREF THAT POINTS DIRECTLY TO ITS ENCLOSING CONCEPT

coreference.[14]

In Figure 12 (the Nexuses are small circles, the Contexts rectangles, and the Description Wires squiggly lines), we have Nexus N1 in Context C1 asserting that a Vulcan named "Spock" is the First Officer of the Enterprise, while in Context C2 these same descriptions are used in a different way by Nexuses N2 and N3 to assert that the First Officer of the Enterprise is a

[14]Co-"reference" is not quite the right term, since the objects "referred to" need not exist. Co-specification of description is probably a better term (see [Sidner 79]).

person named "Uhura" and a Vulcan named Spock is the Captain of the Enterprise. We should note that KL-ONE at the moment does not support any meaningful relations between Contexts, although a hierarchy of Contexts can be created by putting the meta-anchor (i.e., a Nexus) of one Context into another Context.

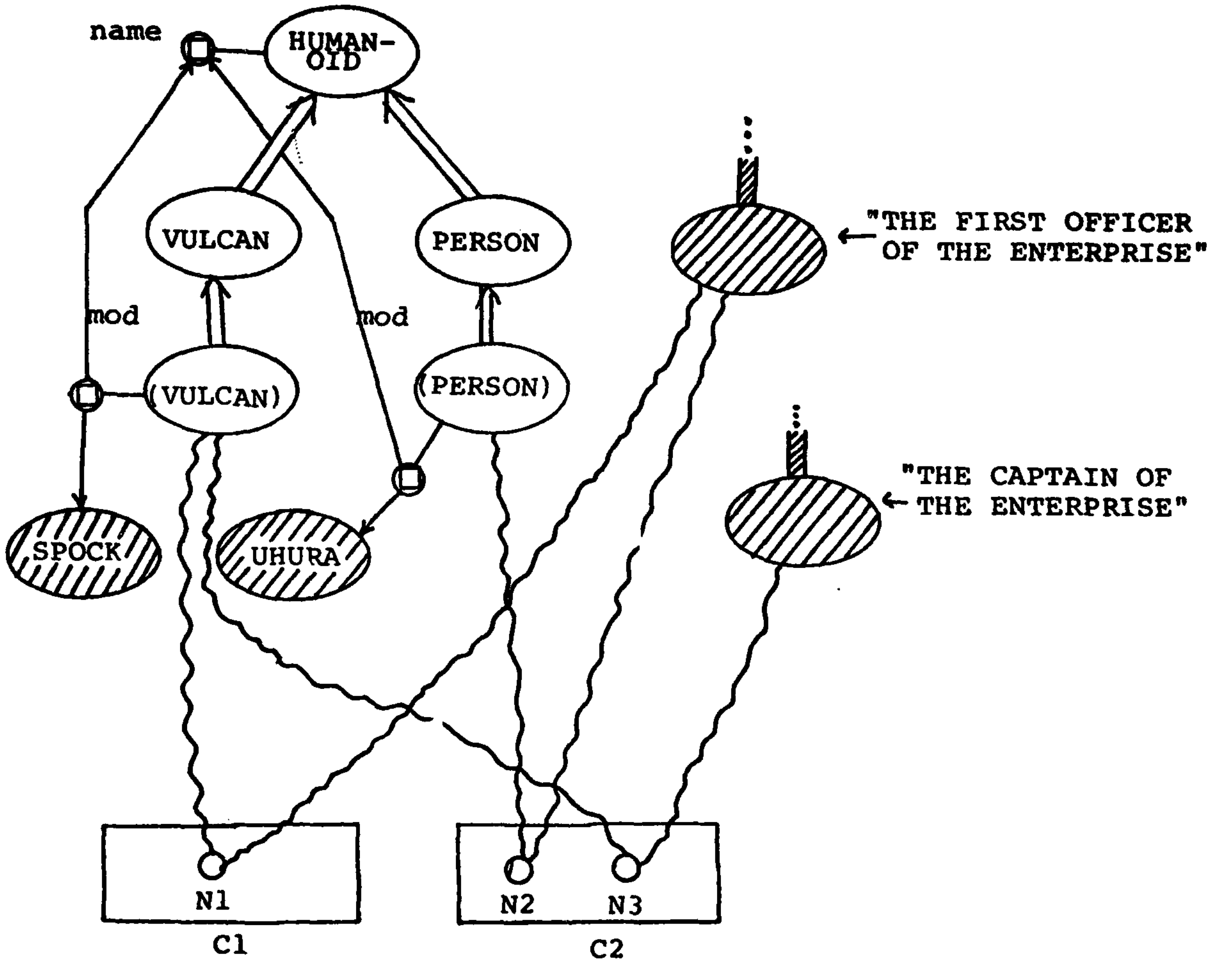

name
HUMAN-OID
VULCAN
PERSON
mod
mod
(VULCAN)
(PERSON)
SPOCK
UHURA
"THE FIRST OFFICER OF THE ENTERPRISE"
"THE CAPTAIN OF THE ENTERPRISE"
N1
N2
N3
C1
C2

FIG. 12. SOME KL-ONE ASSERTIONS.

D.7 Meta-Description

Nexuses allow us to come as close to real "reference" to objects outside the system as is possible in this kind of representation environment. In addition to the use of Nexuses as "surrogates" for outside entities, KL-ONE allows reference to internal entities (e.g., Concepts) as well. Thus one can "meta-describe" a KL-ONE object in KL-ONE. Of course, to do this, the system needs to have the Concepts of a KL-ONE Concept, a KL-ONE Role, a KL-ONE RoleValueMap, etc. These are not yet part of the implemented system.

In order to construct a meta-description, one uses the same type of structure used in constructing a regular description. Each KL-ONE structure is considered implicitly to have a corresponding Nexus that is known to exist in the "KL-ONE base level" Context.[15] Meta-descriptions are simply descriptions (usually expressed in terms of the Concepts KL-ONE-CONCEPT, KL-ONE-ROLE, etc.) attached to those Nexuses by means of the Description Wire mechanism mentioned above. In the future, we expect to study how further to exploit meta-description in KL-ONE. One can imagine the KL-ONE system providing automatic access to a complete meta-description of any other description, and also allowing one to affect the KL-ONE interpreter by modifying these meta-descriptions in a manner similar to that of Brian Smith's 3-LISP [Smith 82]. The primary questions in this effort deal with the details of such a system, and more importantly, the determination of exactly what leverage one gains by using it.

D.8 Attached Procedures and Data

The final feature of KL-ONE to be touched on here is the ability to attach procedures and data to structures in the network. This is purely a programming convenience - attached procedures and data are "outside" of KL-ONE and have no semantically justifiable place in the epistemology. Hence, this section deals strictly with our implementation.

The attached procedure mechanism is implemented in a very general way. Procedures are attached to KL-ONE entities by "interpretive hooks" (ihooks)

---

[15]We have on occasion called these Nexuses "meta-anchors", in the manner of [Smith 78].

(see [Smith 78]), which specify the set of situations in which they are to be triggered. An interpreter function operating on a KL-ONE entity causes the invocation of all procedures inherited by or directly attached to that entity by ihooks whose situations match the intent of that function. Situations include things like "Individuate", "Modify", "Create", "Remove", etc. In addition to a general situation, an ihook specifies when in the execution of the interpreter function it is to be invoked (PRE-, POST-, or WHEN-).

Procedures attached to the conceptual taxonomy can make KL-ONE work like a special kind of object-oriented programming system. We make no claims about this use of the system (but see [Goodwin 79]) - the procedures are not themselves written in KL-ONE, and there can be no guarantee that an attached procedure will honor the integrity of the network. The facility itself is supported only in a very simple way. Attached procedures should ultimately go away when we know how to descriptively characterize behavior in general and its relationship to intensional description.

Finally, a facility has been incorporated to attach arbitrary data to KL-ONE Concepts. The data is stored in property list format and is inherited along superC cables. An second attached data facility exists which simply provides a property list format without inheritance.

REFERENCES

[Brachman 78] Brachman, R. J.
A Structural Paradigm for Representing Knowledge.
Technical Report No. 3605, Bolt Beranek and Newman Inc., May, 1978.

[Brachman 79] Brachman, R. J.
On the Epistemological Status of Semantic Networks.
In Findler, N. V. (editor), Associative Networks: Representation and Use of Knowledge by Computers, pages 3-50. Academic Press, New York, 1979.

[Brachman 80] Brachman, R. J.
"I lied about the trees".
Unpublished manuscript, 1980.

[Fahlman 79] Fahlman, S. E.
NETL: A System for Representing and Using Real-World Knowledge.

M.I.T. Press, Cambridge, MA, 1979.

[Goodwin 79] James W. Goodwin.
Taxonomic Programming with Klone.
Technical Report LiTH-MAT-R-79-5, Informatics Laboratory, Linkoeping University, Linkoeping, Sweden, 1979.

[Hendrix 79] Brachman, R. J.
Encoding Knowledge in Partitioned Networks.
In Findler, N. V. (editor), Associative Networks: Representation and Use of Knowledge by Computers, pages 51-92. Academic Press, New York, 1979.

[Quine 60] Quine, W. V. O.
Word and Object.
The M.I.T. Press, Cambridge, MA, 1960.

[Reiter 80] Reiter, R.
A Logic for Default Reasoning.
Artificial Intelligence 13(1,2):81-132, April, 1980.

[Schubert, et al. 79]
Schubert, L. K., Goebel, R. G., and Cercone, N. J.
The Structure and Organization of a Semantic Net for Comprehension and Inference.
In Findler, N. V. (editor), Associative Networks: Representation and Use of Knowledge by Computers, pages 121-175. Academic Press, New York, 1979.

[Sellars 17] Sellars, R. W.
The Essentials of Logic.
The Riverside Press, Cambridge, MA, 1917.

[Sidner 79] Sidner, C. L.
Towards a Computational Theory of Definite Anaphora Comprehension in English Discourse.
Technical Report AI-TR-537, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, June, 1979.

[Smith 78] Smith, B. C.
Levels, Layers, and Planes: The Framework of a Theory of Knowledge Representation Semantics.
Master's thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, February, 1978.

[Smith 82] Smith, B. C.
Reflections and Semantics in a Procedural Language.

PhD thesis, Artificial Intelligence Laboratory, Massachusetts Institute of Technology, January, 1982.

[Woods 75] Woods, W. A.
What's in a Link: Foundations for Semantic Networks.
In Bobrow, D. G., and Collins, A. (editors), Representation and Understanding: Studies in Cognitive Science, pages 35-82. Academic Press, New York, 1975.

APPENDIX E
INDEX: KL-ONE TECHNICAL TERMS

The following index attempts to bridge the terminological gap that newcomers to KL-ONE must cross by giving brief descriptions of a number of technical terms used in this proceedings. Each description also includes a reference to a more complete explanation within this proceedings. The set of terms included here is, at best, a guess at those terms that a new KL-ONE reader needs to know. Please accept our apoligies for those that we missed.

AETN: Acronym for "Augmented Event Transition Network". See page 64.

Attached Data: Mechanism for attaching arbitrary data to KL-ONE object. See Section D.8.

Attached Procedure: Mechanism for influencing the KL-ONE interpreter. See Section D.8.

CIR: Inverse of the RIC link. See RIC in this index.

CKLONE: Software package for constructing KL-ONE networks. See page 41.

Classifier: A classifier for KL-ONE will automatically place new descriptions at their appropriate location in a KL-ONE taxonomy. See pages 128 and 244.

Concept: Usually refers to all types of KL-ONE Concepts, but it is often used to refer only to Generic Concepts. The Concept is introduced on page 235 and is discussed throughout Appendix E.

Context: Object that contains a set of assertions. See Section D.6.

Coref Wire: Wire that shows the value for a CorefRole. See page 252.

Coref Role: Type of Role found on a ParaIndividual Concept. See page 252.

CorefSatisfies Wire: Wire that shows which RoleSet a CorefRole corresponds to. See page 252.

CorefValue Wire: Synonym for Coref Wire. See Coref Wire in this index.

CSats Wire: Synonym for CorefSatisfies Wire. See CorefSatisfies Wire in this index.

Description Wire: Wire that asserts the correspondence of a Nexus to a Concept. See page 254.

Diff Role: A RoleSet that differentiates another RoleSet. See RoleSet Differentiation in this index.

Differentiates: See RoleSet Differentiation in this index.

ABox: Assertional Component of the proposal by Ron Brachman and Hector Levesque. See page 8.

FranzLisp: Version of Lisp (pattered after MacLisp) that runs on VAX machines. It is the target language of a translation effort for the KL-ONE system. See pages 68 and 106.

Generic Concept: Primary KL-ONE object which represents a general term. Introduced on page 235 and discussed throughout Appendix E.

Generic RoleSet: Represents a conceptual subpart of a Generic Concept. Introduced on page 238 and discussed in Sections D.2 and D.4.

IC: Shorthand for Individual Concept. See Individual Concept in this index.

Ihook: Shorthand for Attached Procedure. See Attached Procedure in this index.

Individual Concept: Object representing a individual term. Introduced on page 235 and discussed in Appendix D.2.

Individual Role: Full name of IRole. See IRole in this index.

Individuator: An Individual Concept. See Individual Concept in this index.

IRole: Represents the binding of a Role, at an Individual Concept, to a description of its filler. Introduced on page 238 discussed in Sections D.2 and D.4.

KL-ONE I/O facility: Input/output facility for KL-ONE networks. See page 38.

KloneTalk: SmallTalk version of KL-ONE written at Xerox-Parc. See page 90.

KLONEUSERS: A directory containing INTERLISP packages that aid a KL-ONE programmer. It is similar in nature to LISPUSERS.

Magic Concepts: A "magic" Concept is one that is incompletely specified viz-a-viz KL-ONE. We have dropped usage of "magic" in favor of "starred". See Starred Concepts in this index.

MetaHook: Mechanism for representing meta-descriptions. MetaHook refers to the Nexus to which a meta-description is attached. See Section D.7.

Meta Link: See MetaHook in this index.

Modifies Wire: Wire representing a Modifies relation between RoleSets. See RoleSet Modification in this index.

Mod RoleSet: A RoleSet that modifies another RoleSet. See RoleSet Modification in this index.

Nexus: Object of co-reference for assertions. See Section 254.

ParaIndividuates Cable: Show the relation between a ParaIndividual Concept and its "parent" Generic Concept. See Paraindividual Concept.

ParaIndividual Concept: Represents a parameterized version of a Generic Concept. See page 252.

Particular RoleSet: Name for a RoleSet that is associated with an Individual Concept. See page 240.

PI: Shorthand for ParaIndividual Concept. See ParaIndividual Concept in this index.

QUA Concept: Representation for a general role-filler, which is an extension of the KL-ONE language. See page 55.

Realizer: A process developed at USC/ISI that attributes new descriptions to individuals as a system learns about them. See page 78.

RIC: Acronym for "role in concept", which is an extension of the KL-ONE language. See page 59.

Role: Represents a conceptual subpart of a Concept, for which there are several types. Introduced on page 238 and discussed in Sections D.2 and D.4.

Role Satisfaction: Represents the binding of a Role to a description of a filler. See Section D.4.

RoleChain: A path through connected RoleSets. See page 252.

RoleName: Each Role can have a name, which is inherited by subRoles. See page 240.

RoleSet: Usually refers to Generic RoleSet, although sometimes to Roles in general. See Generic RoleSet in this index.

RoleSet Differentiation: Represents a partition of a Roleset into subsets. See Section D.4.

RoleSet Modification: Represents additional restrictions on an inherited RoleSet. See Section D.4.

RoleSet Relation: Represents general relationships between RoleSets. See Section D.5.

Role Value Map: A type of RoleSet Relation that represents wither set equality or set inclusion between RoleSets. See Section D.5.

RSR: Shorthand for RoleSet Relation. See RoleSet Relation in this index.

RVM: Shorthand for Role Value Map. See Role Value Map in this index.

Satisfies Wire: Wire representing a Satisfaction relation between a RoleSet and an IRole. See RoleSet Satisfaction in this index.

SD: Shorthand for Structural Description. See Structural Description in this index.

Starred Concepts, Roles, etc: a star is used to indicate an incompletely specified Concept, etc. It is a synonym for "magic" (i.e. a magic Concept), although "starred" is our preferred term. See Section D.3.2 for a description of starred Concepts and the discussion summary of "Individuality in KL-ONE" (Section 1.3, page 27), for other uses of starred objects.

Structural Description: A type of RoleSet Relation that uses a ParaIndividual Concept and represents a relationship between RoleSets of only a single Concept. See Section D.5.

SubC: Shorthand for SubConcept. See SubConcept in this index.

SubConcept: A Generic Concept that is subsumed by another. See Section D.2.

SuperC: Shorthand for SuperConcept. See SuperConcept in this index.

SuperConcept: A Generic Concept that subsumes another. See Section D.2.

TBOX: Terminological Component of the proposal by Ron Brachman and Hector Levesque (see page 8).

Value Restriction: Representation of the restriction of the potential fillers of a Role. Introduced on page 238 and discussed in Section D.2.

APPENDIX F
INDEX: AUTHORS AND CO-AUTHORS

(Note: The following people all contributed to this proceedings, either as authors, co-authors, or as members of a group which presented a paper at the Workshop. The asterisk following the name indicates that they could not be present at the Workshop.)

Official Distribution List

Contract N00014-77-C-0378

	Copies
Defense Documentation Center Cameron Station Alexandria, VA 22314	12
Office of Naval Research Information Systems Program Code 437 Arlington, VA 22217	2
Office of Naval Research Code 200 Arlington, VA 22217	1
Office of Naval Research Code 455 Arlington, VA 22217	1
Office of Naval Research Code 458 Arlington, VA 22217	1
Office of Naval Research Branch Office, Boston 495 Summer Street Boston, MA 02210	1
Office of Naval Research Branch Office, Chicago 536 South Clark Street Chicago, IL 60605	1
Office of Naval Research Branch Office, Pasadena 1030 East Green Street Pasadena, CA 91106	1
Office of Naval Research New York Area Office 715 Broadway - 5th Floor New York, NY 10003	1

Naval Research Laboratory Technical Information Division Code 2627 Washington, D.C. 20380	6
Naval Ocean Systems Center Advanced Software Technology Division Code 5200 San Diego, CA 92152	1
Dr. A.L. Slafkosky Scientific Advisor Commandant of the Marine Corps (Code RD-1) Washington, D.C. 20380	1
Mr. E.H. Gleissner Naval Ship Research & Dev. Center. Computation & Mathematics Dept. Bethesda, MD 20084	1
Capt. Grace M. Hopper NAICOM/MIS Planning Branch (OP-916D) Office of Chief of Naval Operations Washington, D.C. 20350	1
Mr. Kin B. Thompson NAVDAC 33 Washington Navy Yard Washington, D.C. 20374	1
Advanced Research Projects Agency Information Processing Techniques 1400 Wilson Boulevard Arlington, VA 22209	1
Capt. Richard L. Martin, USN Commanding Officer USS Francis Marion (LPA-249) FPO New York 09501	1
Director National Security Agency Attn: R54, Mr. Page Fort G.G. Meade, MD 20755	1
Director National Security Agency Attn: R54, Mr. Glick Fort G.G. Meade, MD 20755	1

Major James R. Kreer Chief, Information Sciences Dept. of the Air Force Air Force Office of Scientific Research European Office of A rospace Research and Development Box 14 FPO New York 09510	1
Dr. Martin Epstein National Library of Medicine Bldg. 38A, 8th Floor Lab 8600 Rockville Pike Bethesda, MD 20209	1